Project Number: DXF 9901

BEOWULF CLUSTER ANALYSIS: THE BRIGHT UTILITY

A Major Qualifying Project Report:

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Jeffrey A. Cilley

_____

Dennis J. Mattoon

_____

Aaron Chandler-Worth

_____

Date: March 9, 2000

Approved:

Professor David Finkel, Major Advisor _____

Professor David Brown, Co-Advisor_____

1. Beowulf
2. Diagnosis
3. Goddard Space Flight Center

Abstract

The goal of this project was to develop a suite of utilities and diagnostic programs that can be executed on a Beowulf Cluster Computer to verify functionality, certify compatibility and identify and help isolate system faults. The suite provides a simple mechanism for novice users to test a cluster. In addition, its modular and extensible design should allow veteran Beowulf users to easily customize the suite to fit their needs.

TABLE OF CONTENTS

1. INTRODUCTION 4

2. A BRIEF HISTORY 5

3. BACKGROUND INFORMATION 7

3.1. BEOWULF 7
3.2. MESSAGE PASSING MODELS 8
3.2.1. MPI 8
3.2.2. PVM 9
3.2.3. Summary 10

4. PROBLEM DESCRIPTION 11

4.1. REQUIREMENTS 11
4.1.1. Requirement 1: The utility must be extensible and easily customizable. 11
4.1.2. Requirement 2: The utility must require a limited amount of user intervention at run time. 12
4.1.3. Requirement 3: The utility must automate the process of software package verification. 12
4.1.4. Requirement 4: The utility must verify functionality of node hardware 12
4.1.5. Requirement 5: The utility had to provide a means of testing the performance of the cluster 13
4.2. SUMMARY 13

5. DESIGN 14

5.1. SYSTEM STRUCTURE 14
5.2. PROGRAM ASSUMPTIONS AND RESTRICTIONS 18
5.3. SUMMARY 18

6. IMPLEMENTATION 19

6.1. THE COMPLETENESS MODULE 24
6.2. THE COMPATIBILITY MODULE. 27
6.3. THE SYSTEM DIAGNOSIS MODULE 27
6.4. THE PERFORMANCE MODULE 30
6.5. SUMMARY 32

7. TESTING 33

8. CONCLUSIONS 39

8.1. FUTURE WORK 39
8.2. PROJECT IMPRESSIONS 40

APPENDIX A: THE BRIGHT USERS MANUAL **41**

APPENDIX B: AN EXAMPLE LOG FILE **64**

REFERENCES 84

1. Introduction

Most supercomputers are far too expensive for most academic and research institutions with limited funding. Beowulf-class systems, which are high performance clusters of commodity hardware assembled to execute parallel applications, offer an alternative that is cost effective and also able to attain supercomputer performance on some applications. Because of their cost effectiveness, Beowulf systems are gaining a wide user base in academic and research institutions. Their growing popularity has increased the need for software that can aid in the configuration and maintenance of Beowulf clusters. The Goddard Space Flight Center (GSFC) is where Beowulf technology was created and GSFC has always been one of its greatest promoters. The goal of this project was to create a utility that could make the task of configuring and maintaining Beowulf clusters easier. This paper will discuss the history of Beowulf technology (chapter 2), some background information necessary for a meaningful understanding of the project (chapter 3), a description of our problem (chapter 4), how we designed and implemented our utility (chapters 5 and 6 respectively), and finally, our testing strategy (chapter 7).

2. A Brief History

The first Beowulf cluster was built in 1994 at the Center of Excellence in Space Data and Information Sciences (CESDIS) located at Goddard Space Flight Center (GSFC) [6]. This first cluster was sponsored by the Earth and Space Science (ESS) project as part of the High Performance Computing and Communications (HPCC) program. The ESS project was trying to determine if massively parallel computers could be used effectively to solve problems that faced the Earth and Space Sciences community [6]. Specifically, it needed a machine that could store 10 gigabytes of data, was less expensive than standard scientific workstations of the time, and that could achieve a peak performance of 1 Gflops (billions of floating point operations per second) [2].

At that time, commercially available systems that could meet the performance requirement were 10 to 20 times too expensive [2]. Thomas Sterling and Don Becker, two scientists at GSFC, were the first to suggest that a machine that met the needs of the ESS project could be built using commodity off the shelf (COTS) parts linked together in parallel configurations. They knew that it would fall short of the performance requirement but it was determined that it would reach a level of performance that was adequate for the needs of the ESS project.

The first cluster, named Wiglaf, used 16 Intel 80486 100 MHz processors connected by channel bonded Ethernet. It was able to maintain a sustained rate of 75 Mflops on certain applications [2]. The next cluster, named Hrothgar, used 16 Pentium class processors and achieved a sustained rate of 280 Mflops. By the end of 1996, Beowulf clusters had been built that could reach a sustained performance of 1 Gflops. In 1998, a Beowulf cluster utilizing the

DEC Alpha family of processors was able to sustain a performance level of 48 Gflops. This was fast enough to earn the rank of 113th on the worlds 500 most powerful computers list [2].

The rapid growth of Beowulf clusters cannot be attributed solely to their speed. There are many other characteristics of Beowulf clusters that make them viable supercomputer alternatives. The next sections will discuss exactly what Beowulf is.

3. Background Information

What follows is a description of the topics necessary for a meaningful understanding of this project.

3.1. Beowulf

The definition of what makes a system a Beowulf varies among those in the scientific computing community. Some people believe that one can only call a system a Beowulf if it is built in the same manner as the original NASA cluster [2]. Others go to the opposite extreme, asserting that any system of workstations running parallel code can be called a Beowulf cluster. The definition that seems to be most widely accepted, and the one we follow, lies between the two extremes.

Beowulf is a multi-computer architecture used for parallel computing. Beowulf systems are collections of personal computers (PCs, referred to as “nodes”) that are interconnected via Ethernet or some other widely available networking technology. Of these interconnected nodes, one is the server node, referred to as the “root”, and the rest, one or more PCs, are the client nodes, referred to as “compute nodes”. Each node in the cluster is built from commercially available off-the-shelf hardware and is trivially reproduceable. As is the case with hardware, Beowulf clusters make use of commodity software as well. The nodes of a Beowulf cluster run one of many freely available, open-source, Unix-like operating systems [2].

To harness system concurrency and make use of Beowulf’s parallel computing capabilities, a cluster needs more than just an operating system and a particular hardware configuration. A layer of logical structure between the programmer and the parallel system resources, one similar in structure to the physical communications layer, is required. This layer

consists of a parallel computation model. This model can take the form of process level parallelism, shared memory, or in our case, message passing.

3.2. Message Passing Models

Message passing models facilitate interaction among sequential processes. These processes run on the nodes of a cluster, one or more to a processor. They communicate through the use of messages passed via the physical network. The application programming interface (API), fundamental to most message passing models, consists primarily of standardized calls to libraries that handle interprocess communication. Message passing models for parallel computation have been widely adopted because of their similarity to the physical attributes of many multiprocessor architectures. And probably the most widely adopted message passing model is MPI.

3.2.1. MPI

MPI, or Message Passing Interface, was released in 1994 after two years in the design phase. The designers of MPI, a working group convened by the Workshop on Standards for Message Passing in a Distributed Memory Environment, made an effort to include the functionality of several other research projects [4]. These include PVM, another message passing model that will be discussed later. MPI's functionality is fairly straightforward.

MPI programs are written in C or Fortran and linked against the MPI libraries. C++ and Fortran90 bindings are also supported. MPI applications run in a multiple instruction multiple data (MIMD) manner. They consist of a number of "normal" processes, running independently in separate, unshared address spaces that communicate through calls to MPI procedures. One common characteristic of MPI applications is that their processes are often the same executable,

each running in its own address space [4]. Although most common, this is not a requirement. It is possible for each of the individual processes of an MPI program to be instances of different executables. In the case of the former, however, one might wonder how one obtains useful parallelism if each process associated with the application is running the same executable. Each process in an MPI application is assigned a rank. A rank is nothing more than a number assigned to each process, distinguishing it from all others. Each process can ask the MPI library for its rank. Using this unique identifier, processes can split up a problem and each can take a piece without any duplication of effort [4]. In this way, MPI enables a collection of computers to be used as a coherent and flexible concurrent computational resource.

3.2.2. PVM

Another message passing model we dealt with during this project, one which the creators of MPI looked to for ideas, is the Parallel Virtual Machine (PVM) model. Its method of operation is a little more complicated than MPI, though it is not as powerful.

Like MPI, PVM programs are written in C and Fortran, and calls to functions provided by the PVM library handle things like process initiation and message transmission and reception. But unlike MPI, PVM programs require the execution of support software on each node PVM processes run on. The support software is a daemon, pvmd3, which runs each machine in a user-configurable pool, also referred to as a virtual machine. A daemon is a program that runs continuously in order to process certain requests made by the users, and pvmd3 is no exception. It handles things like message routing, data conversion for incompatible architectures, and any other tasks necessary for operation in a heterogeneous, network environment. When a user wants to run a PVM application, pvmd3 must be started on each node which is to be included in the virtual machine. Once the daemons are started, the application can be run from any of the

nodes included in the virtual machine. Users have the ability to run multiple PVM applications simultaneously and overlapping virtual machines are permitted [3].

PVM applications most commonly run in a single-instruction multiple-data (SIMD) fashion. Each process executes the same instructions on a small portion of data and then the results are combined. In a way similar to MPI, PVM supports functional parallelism as well. Each PVM process is assigned a different function and they all work on the same set of data [3]. Using either of these two methods, the PVM message passing model presents a unified and general environment for parallel computation.

3.2.3. Summary

The coupling of Beowulf technology and a message passing model for parallel computation, such as MPI or PVM, presents a paradigm for parallel computing which is both powerful and cost effective. Having described Beowulf clusters and message passing, we will now describe the problem tacked by this project, i.e., to develop a utility that would aid in the maintenance and configuration of Beowulf clusters.

4. Problem Description

The goal of this project was to develop a suite of utilities and diagnostic programs that can be executed on a Beowulf cluster computer to verify functionality, certify compatibility and identify and help isolate system faults. This tool will be incorporated into the collection of system software that is being developed for Beowulf cluster computers. The suite provides a simple mechanism for novice users to test a cluster. In addition, its modular and extensible design should allow veteran Beowulf users to easily customize the suite to fit their needs. This project has been designed and implemented in collaboration with Phil Merkey of the Universities Space Research Association (USRA)/CESDIS as part of a project sponsored by WPI and the GSFC.

4.1. Requirements

This section is an outline and explanation of each given requirement. Descriptions of particular methods and specific implementations of these items are given in later sections.

As with any computer system, Beowulf clusters require a certain level of administration. What is unlike other computer systems however, is the type and complexity of the problems Beowulf cluster administrators can be faced with. The following requirements address the problems common to many Beowulf clusters.

4.1.1. Requirement 1: The utility must be extensible and easily customizable.

Beowulf technology is still evolving. Since its creation in 1994, problems associated with Beowulf clusters have been evolving as well. These problems are largely due to the continuous changes in hardware configurations, network technologies and topologies, and in the

needs of the parallel computing community. The first requirement originated from the need to address this problem. The user had to have the ability to add his/her own tests, so that as the user's needs evolved, our utility could evolve as well. The user also needed the ability to remove any of the tests we provided if they did not wish to run them. By making our utility extensible and customizable, we gave it the ability to adapt to the changing needs of its users.

4.1.2. Requirement 2: The utility must require a limited amount of user intervention at run time.

In order for this utility to be a useful maintenance tool it had to be run on a regular basis. It would be very time consuming if the user had to be present for the entire execution. In order to meet this requirement we made the utility run independently of the administrator.

4.1.3. Requirement 3: The utility must automate the process of software package verification.

Another requirement we had to meet dealt with administration difficulties of current clusters. A common problem associated with cluster administration is ensuring that all the nodes of the cluster have specific pieces of software installed. The process of checking the software on the nodes can be tedious and error prone depending on the number of nodes in the Beowulf cluster. By automating this task and taking it out of the hands of the administrator, the process is far less error prone and not nearly as time consuming.

4.1.4. Requirement 4: The utility must verify functionality of node hardware.

When hardware fails in a Beowulf cluster it can be difficult for the administrator to isolate the fault and in some cases difficult to detect the problem in the first place. Those taking care of Beowulf clusters needed a way to test the hardware of each node in their cluster, i.e., hard

disks, memory, and processors. Our utility had to address this by automating the testing of the hardware on each node in a Beowulf cluster to verify its proper function.

4.1.5. Requirement 5: The utility had to provide a means of testing the performance of the cluster.

The last issue we had to address was performance testing. To an administrator, performance data can be useful in a number of ways. In addition to giving him or her the ability to compare two different cluster configurations, it can be used to both test the impact of cluster upgrades and uncover possible impending network hardware failure.

4.2. Summary

There are many issues that need to be taken into consideration when administering a Beowulf cluster computer. Until this project, no other utility addressed the aforementioned requirements. In the next chapter we will discuss the decisions made to accommodate these requirements during development of the utility.

5. Design

This chapter will discuss the overall design of our utility. It will outline how our utility fulfilled each of the requirements that were given to us. There will be sections dealing with the overall system structure, and the assumptions we made when we were designing our utility.

5.1. System Structure

The first design decision that we made was to divide the functionality of the program into modules. By making the underlying structure modular, we made it easy for administrators to add new modules which gives them the ability to extend the capabilities of the utility to fit their system. The modular design also provided administrators with a means of removing any of our tests that they do not need to run.

There are four main tests that we included with our utility. The utility checks for specific software packages on the root node, checks for consistency across the compute nodes, tests hardware throughout the cluster, and tests the performance of the cluster's network and each individual node. We assigned each of these four functions to a different module. There is also a top-level module called the main module that coordinates the four functional modules. The following diagram (Figure 5.1) shows the overall flow of the utility.

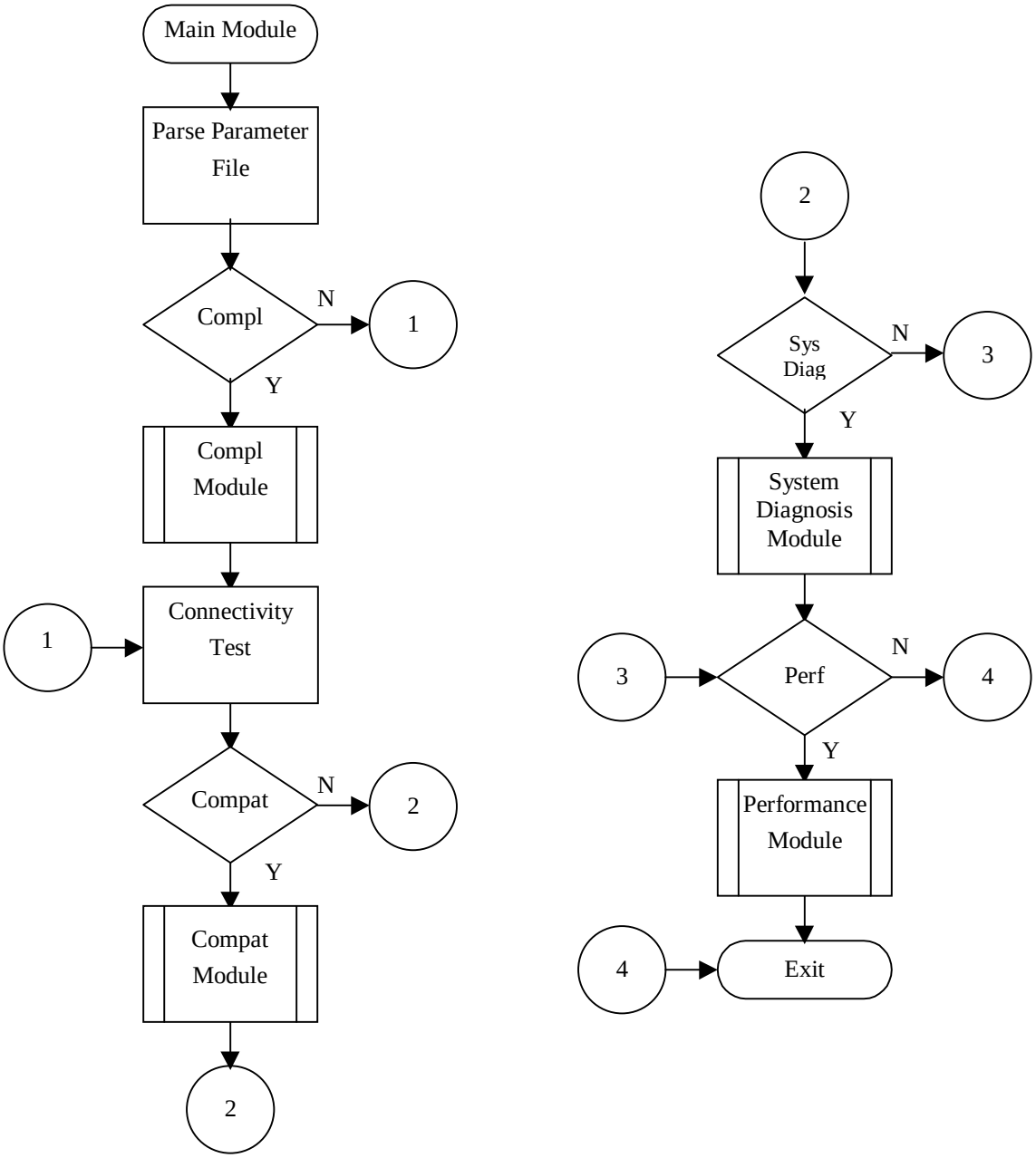

Figure 5.1: Overall flow of the Bright utility

The first module, Completeness, is responsible for checking the software on the root node. The Completeness module receives a list of packages as input from the main module and verifies that each of those packages is installed on the root node. It then generates a list that contains the name of each package and whether or not it is installed on each of the compute nodes. It passes this list back to the main module for handling before it exits.

The second module, Compatibility, is responsible for keeping the configurations of the compute nodes consistent. The main function of the Compatibility module is to execute the Completeness module on all of the compute nodes, thereby ensuring that all the compute nodes have identical software. It is common for the compute nodes to have a different set of software than the root node. The utility needed to allow the user to check for different software on the root and compute nodes. That is why this test was placed in a different module than the software check of the root node. Information from the configuration file, a description of the cluster, is passed to this module from the main module. The Compatibility module passes a list of packages and whether or not they were installed back to the main module once it has completed the tests. Control returns to the main module once all of the compute nodes have finished.

The third module is the System Diagnosis module. It performs hardware tests on all of the compute nodes. This module includes tests of the Central Processing Unit (CPU), main memory modules, and the hard disk drive(s). The CPU test verifies that the CPU is performing calculations correctly. The main memory test checks for corrupted blocks in the memory modules. The hard drive test checks to make sure the disk(s) is not filled to capacity and that the drive(s) is operating normally. Control is returned to the main module once the System Diagnosis module has completed on all of the compute nodes.

The fourth and final module is the Performance module. It is responsible for providing performance information about the system's network and about the computational speed of each of the compute nodes to the administrator. We chose two widely used benchmarking utilities to test each of the nodes and the network. Providing benchmark data for each of the compute nodes allows the administrator to evaluate the effectiveness of hardware upgrades and to help detect hardware failure. The network benchmark can be used to detect failing network hardware and to provide an analysis of a system's network capabilities.

The next requirement that we needed to address was that our utility had to run independently of the administrator once it was executed. This can be difficult because it means that the program needs to be able to handle errors that it encounters without relying on the administrator for input. It also means that the administrator must be able to set all of the options for the utility before it is executed. In order to achieve this, we decided to store all of the input and output of the program in files. There are two input files. The first is the parameter file. It contains a listing of all of the modules, including any that have been added by the administrator, and whether or not those modules should be run. The configuration file contains general information about the cluster. All of the output from the utility is stored in a log file. Any error that occurs will be reported in this file. There are only two errors that will cause the utility to halt. One is if the parameter file cannot be found or is improperly formatted. The other is if the configuration file cannot be found.

Using files for the input and output adds a great deal of flexibility to the utility. First, the administrator can predefine several parameter files with different configurations. This is convenient when there is more than one configuration that is run on a regular basis. Using log

files to store output is useful because it allows the administrator to compare results from previous runs with the current data.

5.2. Program Assumptions and Restrictions

There were a few assumptions we made when we were designing our utility. We assumed that the cluster has the Linux operating system installed. Linux is not the only operating system that is used for Beowulf clusters, but it is common and it is the operating system used at Goddard. We also put a restriction on which distributions could be used. This will be explained in the implementation chapter (section 6.1). The largest restriction that we placed on our utility was that the user's home directory on the cluster had to be NFS mounted. We use temporary files as a means of communication between the nodes and if users had separate home directories on each node, this would not be possible. This again is the configuration that Goddard used and we were told it was common to Beowulf configurations.

5.3. Summary

We feel that our design fulfills the requirements very well. The modular design ensures that the utility is easily customizable and configurable. The use of files for the input and output of the program remove the need for the administrator to be present during the execution of the utility. The few restrictions that we did put on our utility will not exclude many administrators from using it on their system. The tests that we included with the utility should provide administrators with several pieces of useful information without them having to add tests of their own. The next chapter will cover the implementation details of our utility.

6. Implementation

This chapter will discuss how we implemented the modules in our utility to meet the design specifications. It will discuss program input/output and how each module performed its tests. The top-level module in the utility is the main module. Upon execution of the utility, the main module attempts to open the parameter file. The utility expects the file to be in the same directory as the program. It attempts to open the file by executing a system call. If an error occurs when the utility tries to open the file, it typically means that the file is not present. If the parameter file cannot be opened, the utility displays an error message on the screen, prints an error in the log file, and then exits immediately.

Figure 6.1 is an example parameter file. The beginning of every section or subsection in the parameter file is denoted by a tag that consists of the section name surrounded by brackets ([]). The end of the section or subsection is denoted by another tag that is the section name preceded by the string “end_” surrounded by brackets ([]). Each section has a series of statements in the form 'x = y' where x is the parameter and y is its value. The '(yes | no)' notation that appears throughout the example would not be valid in the actual parameter file. It is used here to show that “yes” and “no” are the only two valid options for that particular parameter. The utility will print an error to the screen and exit if the user tries to use an invalid value for one of the parameters.

```
# Comments look like this
[completeness]  # and they can be placed on the same line as entries in the file
  execute=(yes | no)
[end_completeness]

[compatibility]
execute=(yes | no)
[end_compatibility]

[system]
  execute=(yes | no)
  memory_test=(yes | no)
  cpu_test=(yes | no)
  hard_disk_test=(yes | no)
  mem_count=5
  hd_count=2
[end_system]

[performance]
  execute=(yes | no)
  linpack=(yes | no)
  netpipe=(yes | no)
  precision=(single | double)
  rolled=(yes | no)
      [user_defined_test]
      execute = (yes | no)
      path="ls"
      args="-la /home/dmattoon/"
      [end_user_defined_test]
[end_performance]
```

Figure 6.1: Parameter file Format

Every module in the utility has its own section in the parameter file. The example file above has four sections. The four sections are for the modules that we include with the utility. Each of our module sections contains an execute statement which determines whether or not the module will be run. If the execute statement is set equal to "no", the module will not be run and all other arguments for that module will be ignored. If the execute statement is "yes", the remaining arguments will be parsed in order to determine which specific tests the module should run.

At this point, the Completeness and Compatibility modules have no additional arguments other than the execute statement. This will change once we have implemented some of the new

functionality that is listed in the future work section. The System Diagnosis and Performance modules each contain several other parameters that will be explained later in the chapter (Sections 6.3 and 6.4). In Figure 6.1, the section for the Performance module includes a subsection entitled "[user_defined_test]". Any test that the administrator adds must follow the same format as the four main modules. They must begin with a section name surrounded by brackets([]) and they must end with the same section name preceded by the string “end_” surrounded by brackets. There is the standard "execute" statement and then two more arguments that contain the path of the executable and its command line arguments.

Adding a test to the utility is simple. All that the administrator needs is the path of the executable and any command line arguments that need to be given. Any number of tests can be added to any of the modules. It the tests are placed in the completeness module section, they will be run on the root node. If they are placed in any of the other modules’ sections, they will be run on the compute nodes. However, our four modules run in a set order. The location of the results of the user-defined test in the log file will depend on what section they put it in. If the user-defined test is placed in the completeness section, its results will be near the beginning of the log. If the test is placed in the performance section, it will be placed near the end of the log.

There are two different files that store the results from our utility. The first is the log file that contains everything that the modules output. The second file is the summary file. The summary file contains information that we felt was the most important for our modules. It displays the results in a manner that is easier to understand than in the log file. The summary file is also much shorter than the log file. The summary file will not include any information from user-defined tests. In order to view the results of the added test, the administrator has to look through the entire log file. This can be tedious so it is recommended that the user-defined tests

be added to the completeness or performance modules. That way the results will be close the to very beginning or very end of the file.

Once it has been verified that the parameter file is present and formatted correctly, the main module begins executing the other modules in the utility based on the information in the file. The first module that is run is the completeness module. After the completeness module is finished, the main module checks to see which compute nodes are available. The configuration file contains information about the configuration of the cluster. It includes the hostnames and IP addresses of all of the compute nodes and also the list of packages used by the completeness and compatibility modules. Figure 6.2 is an example of a configuration file.

```
# These are the cluster nodes by IP and hostname:
hosts {
      192.168.1.1     h1.hrothgar
      192.168.1.2     h2.hrothgar
      192.168.1.3     h3.hrothgar
      192.168.1.4     h4.hrothgar
      192.168.1.5     h5.hrothgar
      192.168.1.6     h6.hrothgar
      192.168.1.7     h7.hrothgar
      192.168.1.8     h8.hrothgar
      192.168.1.9     h9.hrothgar
      192.168.1.10   h10.hrothgar
      192.168.1.11   h11.hrothgar
      192.168.1.12   h12.hrothgar
      192.168.1.13   h13.hrothgar
      192.168.1.14   h14.hrothgar
      192.168.1.15   h15.hrothgar
      192.168.1.16   h16.hrothgar
}
packages {
      basesystem  tar
      pvfs            tar
      gnu-utils      tar
      pvm            rpm
}
```

Figure 6.2: The configuration file

The example configuration file (Figure 6.2) has two sections. Each section begins with a title and an open brace ('{') and ends with a closing brace ('}'). The hosts section contains the IP

and hostname of each compute node. The packages section contains the names of the software packages that should be installed and also what type each package is. The main module attempts to communicate with each of the nodes listed in the hosts section by using the *ping* command. The *ping* program was originally written by Mike Muuss (pronounced "Moose") for BSD Unix back in 1983. Since that time it has been ported to nearly every platform that exists [7]. It is included in all Unix and Linux distributions. The *ping* program measures the path latency between two systems using ICMP echo packets [7]. Our utility ignores the latency information that *ping* provides. We are merely interested in whether or not the compute nodes are responsive. The utility creates a list of all of the nodes that it was able to reach.

During the initial development, we attempted to use a broadcast address to communicate with all of the nodes at once in order to speed up this portion of the utility. We eventually abandoned this technique and used *ping* on each node individually. The reason we did this was to address compatibility issues. The broadcast *ping* routine worked fine for clusters where there were a small number of compute nodes and all of the compute nodes were on their own private network. It worked perfectly on Hrothgar but when we tried it Ecgtheow, it incorrectly reported nodes as being not responsive when in fact they were. It seemed that *ping* would not receive the responses from all of the nodes when there were a large number of them responding at once. Pinging the nodes individually is slower but more reliable. The utility records any nodes that it was unable to reach in the log file. After the program has a list of reachable nodes, it runs the Compatibility, System Diagnosis, and Performance modules.

6.1. The Completeness Module

The Completeness module is run only on the root node. Figure 6.3 shows the overall flow of the module. It is a separate program that is executed on the root node by the main module.

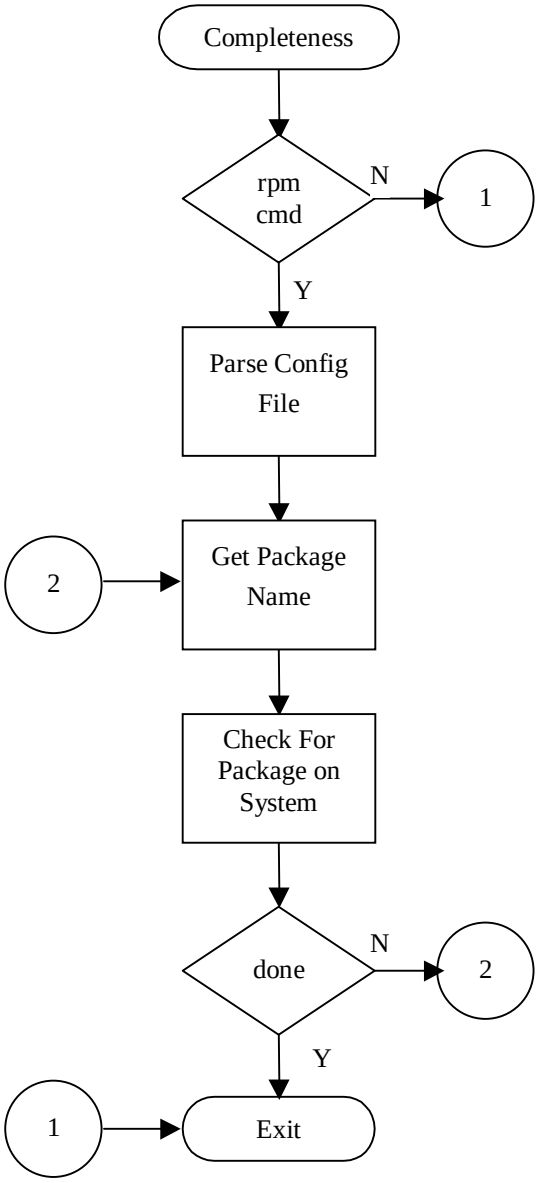

Figure 6.3: Flowchart for the operation of the Completeness module

This module does a check to make sure that the root node has all of the necessary software packages installed. The administrator determines what packages are necessary. He maintains the list software that is in the configuration file (Figure 6.2). The Completeness module uses the *rpm* command to check software on the nodes. *Rpm* is the RedHat Linux package manager software. The *rpm* command can take the source of code of a program and, using spec files (information about the package to be build), generate rpm files that contain the source and/or binaries of the program. Once the program is packaged in the rpm format, it can be distributed and anyone using *rpm* on their system can install it.

The *rpm* utility maintains a database with an entry for each package (rpm file) that it installed. Each entry contains a listing of all of files that were installed for the program. The *rpm* utility can be used to check if a certain package is installed. It will search its database for an entry for that package and report whether the package is installed or not [1]. If the *rpm* software is not present on the system, the module will exit without doing anything. If the *rpm* command is found, the module will use it to verify that every rpm package in the package list is installed on the system. It will report any discrepancies that it finds in the log file. Linux software packages are commonly distributed without executables (source code only) in compressed archives. At this time, our utility does not account for these software packages but there are several resources on the Internet that provide instruction on how to convert these compressed archives to the rpm format.

The Completeness module is also responsible for checking for which compilers are available on the root node. Most Linux distributions include C compilers by default but MPI programs are often written in FORTRAN as well. It is important to know what compilers are

available and what versions they are. Again, the *rpm* command is used to retrieve this information.

6.2. The Compatibility Module.

Once the Completeness module is finished, the utility begins execution of the Compatibility module. This is the first module to be run on the compute nodes. The executable program for this module is different from that of the main module. The main module uses the rsh (remote shell) command to execute the compatibility program on each of the compute nodes. At this time the Compatibility module's only function is to execute the Completeness module's software check on all of the compute nodes. The module stores the results of its execution in a temporary file. The main module waits for the Compatibility module to finish on all of the compute nodes and then it copies the information from the temporary files into the main log file.

6.3. The System Diagnosis module

The System Diagnosis module is responsible for performing hardware tests on the nodes. There are three different tests, a memory test, a central processing unit (CPU) test, and a hard disk test. Each of these tests has its own entry in the System Diagnosis section of the parameter file (figure 6.2). The memory_test, cpu_test, and hard_disk_test parameters are similar to the execute statement, each determines whether or not a particular test will be run. This gives the administrator the ability to customize the module to his needs. Figure 6.5 represents the flow of the System Diagnosis module.

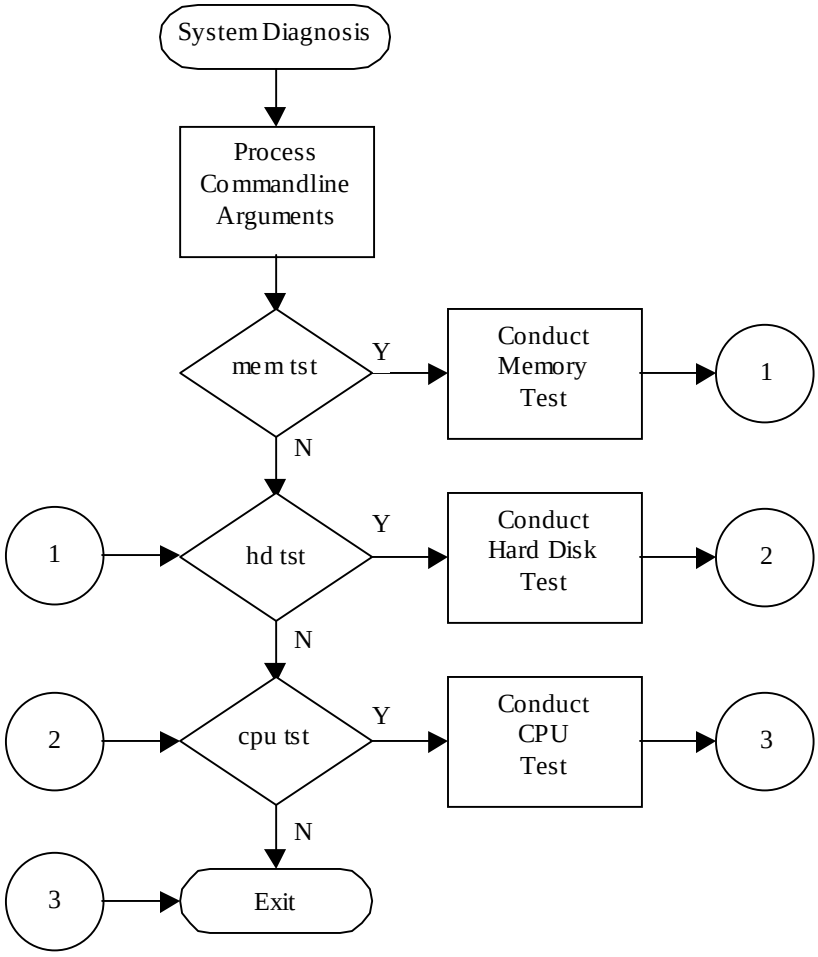

Figure 6.5: Operation of System Diagnosis module.

The memory test is the first test executed by the System Diagnosis module. This test allocates the free memory on the system and writes random bytes of information to it. It then reads all of the information and verifies that it was stored correctly. The utility gets the amount of free memory available by looking at "/proc" directory. On Linux systems, the "/proc"

directory contains information on the system from the kernel. The test uses only the available free memory on the system because we are not interested in the virtual memory. It is possible that the test could report an error when in fact the main memory is working properly if the operating system was forced to store some of the information in virtual memory. In order to test as much of the memory as possible, the memory test allocates several different chunks of memory and performs this read/write verification on all of them. The mem_count value in the parameter file (figure 6.1) determines how many instances of the test will be performed.

The CPU test is the second procedure executed by the system diagnosis module. It exercises the floating-point operations of the CPU. It attempts to determine the accuracy of its calculations. It creates an array of integers and sets every element of the array equal to the inverse of its index. It then proceeds to traverse the array again and multiply each element by its index. The result should be that each element in the array is equal to one. However, floating point operations are not always accurate because there is no way for a computer to represent irrational numbers. This test is able to calculate how close the CPU came to achieving the exact result. This value can be compared to IEEE floating point standards to determine if the result was in fact correct. It should be observed that a processor's precision should not vary once it has been constructed; if a deviation does arise, something is most likely seriously wrong with the processor.

The final procedure run by the System Diagnosis module is the hard disk test. The main function of this test is to determine whether the disk drives on the compute nodes are nearing capacity. It does this by capturing output from the Operating System's *df* command and reporting the percentage of the hard disk that is being used. This test alleviates the burden of having to check the disks of each node individually. It will also prevent errors that can be caused

by having a disk that is completely full. This test also attempts to identify failing disks by timing a series of read/write operations and comparing this result with previous runs. Slow performance is often a symptom of a failing hard disk, this procedure helps find failing drives by catching these symptoms early. The speed of the drives is determined by first allocating enough memory to force the operating system to utilize the system's swap space (virtual memory). It them executes several reads and writes to the disk and times how long it take to perform the entire operation.

6.4. The Performance module

The purpose of the Performance module is to provide a benchmark for the network and the individual nodes of a Beowulf cluster. The two benchmarks that we included with our utility for this purpose are Netpipe and Linpack.

Linpack is a linear algebra package that performs mathematical operations on the CPU and times its performance. The original FORTAN routines were written by Jack Dongarra in the 1970's. It has since been translated into C [8]. There were many reasons why we included Linpack. The first is that our mentor requested that it be included with the utility. The second reason is that it has been around for a while and is very popular. Another reason is that Linpack was originally designed to gauge the speed of super computers which has made it widely used in the Beowulf community.

There are two different options for Linpack in the parameter file (figure 6.1). The 'precision' parameter has two possible values, "single" and "double". If it is set to single, Linpack will perform its operations using single precision floating point numbers. If it is set to double, it will use double precision floating point numbers. The second parameter, rolled, also has two possible values, "yes" and "no". The rolled parameter influences the way the Linpack

utility is run by changing the way its program loops are executed. If rolled is set to “no” then unrolled versions of some program loops are used. This means that some sequences of instructions are not executed within loops. If rolled is set to “yes” then these instructions are executed within loops. Unrolled versions of code are used to avoid the overhead of loop management. In more recent machines and compilers however, there is a higher performance penalty associated with unrolled code than there is with rolled code.

Linpack is run on each individual node to give the administrator an idea of how his/her nodes compare to each other. It can also be used to compare the speed of two different cluster configurations which gives the administrator a means of evaluating the effectiveness of hardware upgrades. All of the output from Linpack is stored in a temporary file that is that the main module adds to the log file once the performance module is completed.

Netpipe was originally developed at Iowa State University. In some distributed systems, including some Beowulf clusters, the network causes the largest performance bottleneck. Netpipe was written specifically to address this issue [5]. Netpipe has been implemented using PVM, MPI, and standard UNIX tcp sockets [5]. It transfers data of varying size between two nodes in order to determine different characteristics of the network such as its speed, throughput, and saturation point. This information can be used to assess the speed of various pieces of network hardware. For example, in order to assess the speed of an Ethernet switch, you could run this benchmark between to computers that are connected directly to each other and then run it again between the same two computers while they are connected through the switch. By comparing the results, you could determine how efficient the switch is. If you did this test with multiple switches, it would allow you to determine which one performed the best.

There are two parameters for Netpipe in the parameter file. The first is the 'execute' statement that has the same functionality that it had in the previous sections. The second is the 'net_count' parameter. This determines how many times Netpipe will be executed. Every time it is executed, it will be run on a different pair of nodes. If the nodes all have the same hardware, the results from each of the runs should be similar. If there is a discrepancy, it might mean that some of the nodes have out-dated hardware drivers or that the hardware is actually failing. Netpipe stores its data in its own file. The user can use programs such as gnuplot (a freely available plotting program) or Microsoft Excel to analyze the results. The Netpipe documentation that was included in its distribution and that we include with our utility gives a detailed description of how to do this.

6.5. Summary

In conclusion, we feel that we successfully implemented all of our design features. The Completeness and Compatibility modules verify that software across all of the nodes is consistent. The System Diagnosis module gives the administrator a tool that will find hardware faults on the cluster. The Performance module analyzes network statistics and the performance of each individual node of the cluster for the administrator.

7. Testing

There were three different Beowulf clusters on which to test Bright. Two of them were located at Goddard. The third was located at WPI. The two Goddard clusters we used, Hrothgar and Ecgtheow, are both compliant with the Goddard definition of what constitutes a Beowulf cluster. They each consist of one root node and several homogeneous compute nodes. Hrothgar has 16 compute nodes. Each compute node has a Pentium processor running at 75 Mhz (Megahertz) and 32 MB (Megabytes) of memory. Ecgtheow has 64 compute nodes. Each node has a Pentium pro processor running at 200 Mhz and 128 MB of memory. The utility was able to successfully complete all of tests on the Goddard clusters.

The cluster at WPI is not compliant with the Goddard definition of Beowulf clusters. It consists of four nodes that each have the DEC Alpha processor. There is no set root node for this cluster, but it did have one central mount point for the home directories so we were able to run our utility on the cluster. In order to run it, we chose an arbitrary node as the root and listed the remaining nodes in the 'hosts' section of the configuration file which makes the utility treat them as compute nodes. The utility was also able to run all of the tests on this cluster.

Our utility was designed to detect faults in a Beowulf clusters and, unfortunately, we didn't have any faulty clusters to test it on. Our mentor informed us that the clusters were configured properly as far as software was concerned. Our utility was unable to find any faulty hardware on the clusters. We therefore had to attempt to simulate the types of failures that our utility was designed to find. These tests were performed only on the Hrothgar cluster because of the two Goddard clusters, Hrothgar was the oldest and least frequently used. We wanted to limit the amount of inconvenience incurred by the users of the Goddard clusters if our utility had caused any problems during its development

The first module that was tested was the Completeness module. We installed several packages and listed them in the configuration file. We also listed several packages that we knew were not installed. The main functionality of the Completeness module relies on the *rpm* utility so the main goal of this test was to verify that the module was correctly interpreting the output that *rpm* produced. The test consisted of listing a variety of packages in the configuration file and verifying in each case that our utility correctly reported whether they were installed or not.

There were three different cases that we used in this test. In the first case, we listed a number of packages that we had personally installed using the *rpm* command. In the second case, we used a combination of packages installed using *rpm* and packages installed not using *rpm*. There were also packages that we listed in the configuration file that did not exist at all. Again, we installed the packages ourselves so we knew for sure which packages our utility was supposed to report as being installed. In the third case we used a combination of packages not installed with *rpm* and a number of packages that did not exist on the system at all. In each test case, our utility correctly reported that a package was installed if it was installed using *rpm*. Any other packages were listed as not installed which is the correct outcome for this module. Figure 7.1 is an example of the output from one of the test cases. Appendix B contains an example of a complete log file. In this case, the first package was installed without the *rpm* command and the second package was installed with the *rpm* command.

```
bright: completeness: Module started on hrothgar Mon Dec 13 19:25:43 1999
bright: completeness: Temporary file bright-compl.hrothgar created in /home/dmattoon/.bright
bright: completeness: Looking for path to rpm command
bright: completeness: Command found, path: /bin/rpm
bright: completeness: Parsing config file for package list
bright: completeness: Package list found
bright: completeness: Checking for pvm package
bright: completeness: Warning: pvm is not installed
bright: completeness: Checking for ssh package
bright: completeness: ssh is installed, packages: ssh-1.2.26-2
bright: completeness: Module finished Mon Dec 13 19:25:43 1999
```

Figure 7.1: Output of the completeness module in the log file

Currently, the Compatibility module is virtually identical to the Completeness module. We only needed to check that its information was being stored in the log correctly. We used the same test cases that we used for the Completeness module. The Compatibility module was able to perform each test on each of the nodes correctly. Figure 7.2 is an example of the results from the Compatibility module on one node.

```
bright: compatibility: h5: Module started on Mon Dec 13 19:07:58 1999
bright: compatibility: h5: Temporary file bright-compat.h5 created in /home/dmattoon/.bright
bright: compatibility: h5: Deleting temporary file /home/dmattoon/.bright/bright-compl.h5
bright: compatibility: h5: Temporary file bright-compl.h5 created in /home/dmattoon/.bright
bright: compatibility: h5: Looking for path to rpm command
bright: compatibility: h5: Command found, path: /bin/rpm
bright: compatibility: h5: Parsing config file for package list
bright: compatibility: h5: Package list found
bright: compatibility: h5: Checking for pvm package
bright: compatibility: h5: Warning: pvm is not installed
bright: compatibility: h5: Checking for ssh package
bright: compatibility: h5: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h5: Module finished Mon Dec 13 19:07:59 1999
```

Figure 7.2: Compatibility module results on node h5

The next step was to verify that the main module was using *ping* to accurately determine whether the compute nodes were responsive. In this test we simulated failure of the network hardware on the nodes by disconnecting them from the cluster. We did several runs of the test.

On each run, we disconnected different nodes. The utility was able to correctly determine which nodes where connected.

Testing the System Diagnosis module was complicated. The memory test was designed to detect faulty memory modules but when a memory module goes bad it is usually removed and discarded so we didn't have access to any for our test. We were also unable to effectively verify the functionality of the CPU test because we did not have access to any faulty CPU's. In the final release of the Bright utility, the CPU test will probably be omitted. We have discovered through our testing that it does not seem to be very useful. A CPU's precision in calculations will be determined by whatever standards, such as the The Institute of Electrical and Electronics Engineers(IEEE) standards, it was designed to follow. Also, if a processor does fail, the system will probably not boot at all which will be good sign that there is something wrong. The hard disk test worked very well on all of the clusters. It successfully output warnings when the local disk was nearing its capacity.

The hard disk test was also designed to find any faulty hard drives but like the memory test, we didn't have any faulty hardware to test it on. We know from personal experiences that hard drives will tend to have slower performance before failing completely so the test should work. Figure7.3 shows the output from the system diagnosis module on Ecgtheow. The test did not find any errors when it ran the memory test or the hard disk test. It also did not find any errors in the CPU test.

```
bright: sysdiag: e1: Module started
bright: sysdiag: e1: Disk Information:
bright: sysdiag: e1: Filesystem           Size  Used Avail Use% Mounted on
bright: sysdiag: e1: /dev/hda1            991M  106M  833M  11% /
bright: sysdiag: e1: /dev/hda3            1.9G   13k  1.8G   0% /scratch
bright: sysdiag: e1: manager:/home        2.8G  1.7G  919M  66% /home
bright: sysdiag: e1: manager:/            2.9G  1.1G  1.6G  41% /manager
bright: sysdiag: e1: Memory Information:
bright: sysdiag: e1:   Total:    Used:    Free:  Shared: Buffers:  Cached:
bright: sysdiag: e1: 131231744 25321472  105910272 6565888  9830400 10125312
bright: sysdiag: e1: Beginning memory testing
bright: sysdiag: e1: Allocating 105910272 bytes
bright: sysdiag: e1: Memory test completed successfully!
bright: sysdiag: e1: Beginning cpu testing
bright: sysdiag: e1: Accurate to 20 digits in calculations
bright: sysdiag: e1: Accurate to 20 digits in memory
bright: sysdiag: e1: Precise to 17 decimal places
bright: sysdiag: e1: Beginning hard disk testing
bright: sysdiag: e1: allocating 65615872, 65615872, and 65615872  bytes
bright: sysdiag: e1: hard disk test took 19350000msec to complete
bright: sysdiag: e1: Hard disk test completed successfully!
bright: sysdiag: e1: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
```

Figure 7.3: System Diagnosis output on node e1

The performance module was the last module that had to be checked. We did not do much testing of the functionality of Linpack and Netpipe. Both of these benchmarks are widely used so we assumed that they worked properly. The only challenge was integrating the output from the benchmarks into our log file.

We also tested several user-defined tests in our utility. The purpose of these tests was to verify that user-defined tests were being executed on the proper node. We placed several user-defined tests in each of the four main module sections. When the test was placed in the section for the Completeness module, the utility executed the test on the root node. When the test was placed in the other modules, it was executed on the compute nodes. In each module, the tests executed correctly.

In general, we feel that our utility performed well. At the very least, it will provide administrators with a framework that they can use to combine their own diagnostic utilities into one package.

8. Conclusions

The goal of this project was to create a utility that would simplify configuration, by automating the task of verifying that all the necessary software is installed, and aid in the maintenance of Beowulf clusters. Our suite provides a simple mechanism for novice users to test a cluster. In addition, its modular and extensible design gives veteran Beowulf users the ability to easily customize the suite to fit their needs. We feel that our utility not only performs the functions it was designed for but also provides a useful framework for administrators to build on or use in the design of their own utilities.

8.1. Future Work

We feel that more could be added to our utility before it is officially released to the Beowulf community. Currently our utility is specialized somewhat to certain types of Beowulf clusters, i.e., clusters that are homogeneous in nature and have one centralized access point for home directories. This requirement could be softened by modifying the behavior of some of the control scripts and modules.

Another addition our program could benefit from is a password file concurrency check. This should consist of a program or script that compares the password file on the root node to the client nodes and reports which nodes are not consistent with the root.

The performance module could also benefit from further work. It could be modified to automatically generate the graphs that represent network characteristics such as a network's saturation point or Ethernet signature. An alternative to this could be some sort of textual representation of these figures in the log file or summary file. There are numerous ways to extend the functionality of this utility and probably many that we have not thought of. This is

one of the reasons why we intend to release our utility to the Beowulf community; so that those in the community can use the utility and provide us with feedback.

8.2. Project Impressions

Our overall impressions of the project are good. We enjoyed working with our Mentor, Phil Merkey. We also enjoyed working at Goddard Space Flight Center. It gave us an excellent opportunity to learn about Beowulf technology. The only complaint the we have is that we feel the project was disorganized in nature. We did not have a clear goal for project before we arrived at Goddard. It took us a couple weeks in the beginning to really get started. Other than that, we have no complaints. We would recommend the Goddard project center to anyone interested in doing off-campus project work.

Appendix A: The Bright Users Manual

BRIGHT

USER'S MANUAL

TABLE OF CONTENTS

APPENDIX A: THE BRIGHT USERS MANUAL **41**

1. INTRODUCTION **43**

2. INSTALLATION AND CONFIGURATION **43**

2.1. CLUSTER REQUIREMENTS 43
2.2. INSTALLATION 43
2.3. CONFIGURATION 44
2.3.1. The Configuration File *44*
2.3.2. Configuration File Customization *47*
2.3.3. The Parameter File *47*
2.3.4. Adding Your Own Module *50*

3. THE MODULES **52**

3.1. WHAT EACH MODULE DOES 52
3.1.1. Completeness Module *52*
3.1.2. Compatibility Module *53*
3.1.3. System Diagnosis Module *53*
3.1.4. Performance Module *56*

4. USING BRIGHT **58**

4.1. RUNNING BRIGHT 58
4.2. INTERPRETING THE RESULTS 59

5. CONCLUSIONS **63**

5.1. CONTACT INFORMATION 63

1. Introduction

Bright is a suite of utilities for the analysis, diagnosis, and performance evaluation of Linux-based Beowulf Clusters. Its goal is to verify functionality, certify compatibility and identify and help isolate system faults. This suite provides a simple mechanism for novice administrators to test a cluster. In addition, its modular and extensible design allows veteran Beowulf users to easily customize the suite to fit their needs.

This manual describes version 0.99 of Bright. This is the WPI MQP version. If and when this utility is released, this manual will be updated.

2. Installation and Configuration

This section will discuss the issues surrounding the installation and configuration of the Bright utility.

2.1. Cluster Requirements

Bright was designed to run on as many cluster configurations as possible. There are relatively few requirements placed on a cluster. Each node needs to have the users' home directories mounted, usually /home, from a common point, which is usually the root node. It also requires that the Linux installation used makes use of the /proc file system (most do) and that Perl and the Red Hat Package Management system be installed.

2.2. Installation

Currently, a user with normal user privileges can run the Bright utility. The archive bright-0.99.tgz needs to be extracted somewhere in /home, such as in the user's home directory. The command to extract the file is 'tar xvfz bright-0.99.tgz'. This will create a directory called

bright with all of the program files. Once extracted, the utility needs to be set up to run on the cluster. The next section will describe the configuration process.

2.3. Configuration

This section will walk you through the process of configuring the Bright utility. It will begin by discussing the configuration file and its uses. This chapter will then explain the parameter file beowulf.param, what needs to be done to get tests to run and how the user can add his or her own tests.

2.3.1. The Configuration File

The configuration file, called beowulf.conf, stores information about the user's Beowulf cluster. Figure 1 shows an example file.

```
# Generated by the Beowulf Cluster Configuration Program

# These are the cluster nodes by IP and hostname:
hosts {
    192.168.1.1     h1.hrothgar
    192.168.1.2     h2.hrothgar
    192.168.1.3     h3.hrothgar
    192.168.1.4     h4.hrothgar
    192.168.1.5     h5.hrothgar
    192.168.1.6     h6.hrothgar
    192.168.1.7     h7.hrothgar
    192.168.1.8     h8.hrothgar
    192.168.1.9     h9.hrothgar
    192.168.1.10    h10.hrothgar
    192.168.1.11    h11.hrothgar
    192.168.1.12    h12.hrothgar
    192.168.1.13    h13.hrothgar
    192.168.1.14    h14.hrothgar
    192.168.1.15    h15.hrothgar
    192.168.1.16    h16.hrothgar
}
basehostname h
basedomainname hrothgar

# Ethernet information for the head node
headIP 192.168.1.17
headDEV eth0

# What boot image to use
image boot_tulip.img

# Root password (encrypted)
rootpass "BLANK FOR OBVIOUS REASONS"

# Partition mount points, size, and growable option
partitions {
    swap 64
    /  1000 grow

}
# NFS exports and mount points (left is head, right is node)
enablenfs no

install_loc /usr/b_image

packages {
    basesystem      tar
    pvfs            tar
    gnu-utils       tar
    pvm             rpm
    mpich           rpm
}
```

Figure 1: Sample beowulf.conf file

Figure 1 shows a sample taken from one of the Beowulf clusters at the Goddard Space Flight Center. The beowulf.conf file can come from one of two places. The user can either edit the one that is included with the Bright utility or he or she can use one generated by a separate utility. This other utility is one that automates the installation of Beowulf clusters and is currently in development at Clemson University. As you can see in the first line of the sample in Figure 1, this sample configuration file was generated with that utility. If you use the Beowulf installation utility it is recommended that you use the configuration file that it generated. However, this is not essential.

Figure 2 shows the configuration file that is included with the Bright utility.

```
#Beowulf configuration file: /etc/beowulf.conf

# This section describes the compute nodes of the cluster
# The format for this section is <ip address>  <hostname>
# Example:
# hosts {
#       192.168.1.1             node1
#       192.168.1.2             node2
#  }
hosts{
}

# This section describes the packages that are
# expected to be installed across the cluster
#
# Currently Bright will only check for the presence of rpms
# in future versions we hope to include support for other
# package/archive types.
#
# The format for this section is <package name>  <archive type>
# Example:
# packages {
#     basesystem        tar
#     pvfs                      tar
#     gnu-utilstar
#     pvm                       rpm
#     mpich                     rpm
#}
packages {
}
```

Figure 2: Configuration file distributed with Bright.

To create and use your own configuration file you need to fill in the host and package information sections of the configuration file included with Bright. The following section covers customization of the configuration file.

2.3.2. Configuration File Customization

The hosts section of the configuration file needs to be filled in with the IP addresses and names of each node in your cluster, using the sample provided as an example. At present, the hostnames are not used but are included in order to remain compatible with the installation utility. The packages section needs to be maintained by the cluster administrator, or the user, and must include package names followed by archive type (Figure 1). The package list will be discussed further in section 3.1.1. Currently, package types other than rpm are ignored but in future versions we hope to include support for different archive types. Once modified, the beowulf.conf file needs to reside in the /etc directory on the root node.

2.3.3. The Parameter File

The parameter file, beowulf.param, controls the behavior of the Bright utility. It should be located in in /home/<user>/bright-0.99/modules/main where <user> is the username of the person running the utility. Figure 3 shows the parameter file that is distributed with Bright.

```
#Bright parameter file

[completeness]
   execute=yes
[end_completeness]

[compatibility]
  execute=yes
[end_compatibility]

[system]
  execute=yes
  memory_test=yes
  cpu_test=yes
  hard_disk_test=yes
  mem_count=1
  hd_count=1
[end_system]

[performance]
  execute=no
  linpack = yes
  netpipe = yes
  precision = double
  rolled = no
  net_count = 1
 [end_performance]
```

Figure 3: Parameter file distributed with Bright.

The parameter file has four sections, one for each of the modules, Completeness, Compatibility, System Diagnosis, (abbreviated to System) and Performance. Each module can also have subsections defining the behavior of user-defined modules, adding user-defined-modules will be covered section 2.3.4. Different module parameters are set within each section. To see exactly how they work we will describe the settings for one of the modules. We'll go through the System Diagnosis module because it is currently one of the largest modules in the utility.

Figure 4 shows the System Diagnosis section of the parameter file.

```
[system]
  execute=yes
  memory_test=yes
  cpu_test=yes
  hard_disk_test=yes
  mem_count=1
  hd_count=1
[end_system]
```

Figure 4: Parameters for the System Diagnosis module

Each module in the parameter file begins with the module's name enclosed in square brackets (this includes user-defined modules which will be discussed in section 2.3.4). In the case of the System Diagnosis module in Figure 4, this has been abbreviated to [system]. Each section must begin with the proper heading. The four valid headings are [completeness], [compatibility], [system], [performance]. Each module's section in the parameter file contains information necessary for that module to run.

The first item of this section is the execute parameter. The presence of the execute parameter is necessary in every module's parameter file section. In our example, setting this parameter to "yes" means that the System Diagnosis module will be included when Bright is run, meaning that the System Diagnosis module will be executed on each node in the cluster. If the execute flag is set to "no" this module will, of course, not be executed.

The following three flags memory_test, cpu_test, and hard_disk_test are similar to the execute parameter in that their values will determine whether the tests they are named for will run or not. If memory_test is set to "yes" then that portion of the System Diagnosis module will be included when the module is run. The same is true for cpu_test and hard_disk_test. What come next are the parameters for the memory and hard disk portions of the System Diagnosis module. These values, mem_count and hd_count, must be integers. They indicate the number of

times the memory and hard disk tests will be run on each node. If the execute flag is set to “no” then all other flags in the module's parameter file section are ignored.

Once all necessary parameters are represented, the ending of the module's section is signified by a line containing that module's name preceded by "end_". This is true for all modules, and in our example the end of System Diagnosis section is represented by [end_system].

Now, with an understanding of the basic layout of a module's entry in the parameter file we will discuss adding a user-defined module. Each module and each parameter in the parameter file will be discussed in detail in section 3.1.

2.3.4. Adding Your Own Module

As Beowulf clustering technology evolves, so do the requirements placed on the administrator. Hence, our utility was designed to be extensible, by giving the user the ability to add his or her own modules, so that Bright can adapt to the users' changing needs. This is one of the most important features of Bright.

Adding a module is a relatively simple process. Before adding a module you first need to decide where in the execution of Bright you want it to be run. A module can be inserted in any one of the default module sections of the parameter file. The module section in which a user-defined module's parameters are listed will be the module that is run just before the user-defined test. This gives the user the ability to run their module at any point during the execution of Bright. Figure 5 illustrates the format of a user-defined module if it were to appear within the performance section of the parameter file.

```
[performance]
  execute=yes
  linpack = yes
  netpipe = yes
  precision = double
  rolled = no
  net_count = 1
  [my_test]
        execute = no
        path = "/usr/bin/my_test"
        args = "-my -args"
  [end_my_test]
[end_performance]
```

Figure 5: Sample parameter file

Because the user-defined test my_test appears within the performance module, it will be run after the performance module has completed. Say a user-defined test were inserted in the Compatibility section of the parameter file, it would be executed after the Compatibility module, but before the System Diagnosis module. If the module is placed in the section for the Completeness module it will be run only on the root node. If it is placed in any of the other sections, it will be run on the compute nodes.

The parameter file section describing the user-defined test can appear at any point within a default module's section, even before the modules execute flag is set. And, just as with other flags in a module's parameter file section, if the module's execute flag is set to no, any user-defined module descriptions appearing within the default module's description are ignored. What follows is a description of the parameters which need to be included in a user-defined module's parameter file section.

In a user-defined test there are three parameters execute, path, and args. As you could probably guess the execute parameter determines whether or not this user-defined module is included in the execution of Bright; its function is identical to that of the execute flag in a default module. Next comes the path parameter. This indicates the path to the executable that the user

wants to run in this module. Finally, the last parameter args needs to be set to a string containing all the arguments to be passed to the program described by the path parameter. As shown in the example in Figure 5, user-defined modules need to have the same type of structure as the default modules, meaning that they begin with the module's name enclosed in brackets, in this case [my_test], and end with the phrase "end_" preceding the modules name enclosed in brackets, as in [end_my_test].

3. The Modules

This chapter gives a description of each module and it also lists every parameter that each module has in the parameter file.

3.1. What Each Module Does

Bright is made up of four main modules: Completeness, Compatibility, System Diagnosis, and Performance. Even though the modules run in sequence, any module or modules the administrator feels he or she can do without, can be left out of an execution of Bright. The following section describes just what the modules do, once they are described, there will be a section about actually using the utility.

3.1.1. Completeness Module

The Completeness module is responsible for verifying the presence of all necessary software packages on the root node. The execution of the Completeness module does not affect any of the compute nodes. This module uses the package list in the configuration file that was described in a previous section. The package list needs to be maintained by the user or cluster administrator and should contain the names of packages, specifically Red Hat packages (rpms), that are

supposed to be installed on the root node. The following table shows the parameters for the Completeness module.

Parameter	Values	Description
execute	yes / no	Whether to execute the module

3.1.2. Compatibility Module

The Compatibility module is run on each compute node and its job is to ensure that all of the nodes have the same version of each software package that the root node has. This list of necessary packages is the same one used in the Completeness module; it is the list of packages found in the configuration file. The parameter file entries for this module and the Completeness module are very small, each only contains the execute flag. The following table shows the parameters for the Compatibility module.

Parameter	Values	Description
execute	yes / no	Whether to execute the module

3.1.3. System Diagnosis Module

The job of the System Diagnosis module is to report hardware errors, possible warning signs of impending hardware failure and general hardware status. This module is also run on each compute node. This module performs tests on cluster memory, hard disks, and CPUs. Each of the three tests has its own parameters associated with them in the System Diagnosis section of the parameter file.

3.1.3.1. The Memory Test

First, we will discuss the memory test. The purpose of this test, as the name implies, is to test system memory. The test itself was adapted from a utility called Lucifer originally released by Peter Todd in June of 1999 [9]. This portion of the System Diagnosis module will perform a burn-in test meaning it will write a random value to each byte in free memory and read it back from memory. If the value read does not match the value written then there is a problem with that memory module and the user is notified. There are two parameters associated with this portion of the System Diagnosis module memory_test, and mem_count. The value of the memory_test parameter determines whether the test is run or not. The mem_count parameter specifies the number of times the memory test will execute.

3.1.3.2. The CPU Test

The CPU test is the second procedure executed by the system diagnosis module. It exercises the floating-point operations of the CPU and attempts to determine the accuracy of its calculations. It creates an array of integers and sets every element of the array equal to the inverse of its index. It then proceeds to traverse the array again and multiply each element by its index. The result should be that each element in the array is equal to one. However, floating-point operations are not always accurate because there is no way for a computer to represent irrational numbers. This test is able to calculate how close the CPU came to achieving the exact result. This value can be compared to IEEE floating point standards to determine if the result was in fact correct. It should be observed that a processor's precision should not vary once it has been constructed. If even a small deviation does arise, something is most likely seriously wrong

with the processor. This test takes only one parameter from the System Diagnosis section of the parameter file, the cpu_test parameter. It determines whether or not the CPU test will be run.

3.1.3.3. The Hard Disk Test

Finally, we will discuss the hard disk portion of the System Diagnosis module. The purpose of this test is to make sure the hard disks of the nodes are not showing signs of impending failure. This test is conducted in a manner similar to the memory test. Three chunks of memory are allocated and the size of each chunk is one half the size of total system memory. Because more memory is allocated than there is physical memory on the system the swap space will be used. Random bytes are written to memory and later read back and verified against the original values, just like the memory test. But what is different from the memory test is that the hard disk test is timed. Some drive errors may not be noticed because of error correcting mechanisms within the drive so the test is timed because failing drives often take inordinate amounts of time to complete relatively small tasks. By reporting these large times Bright helps administrators identify failing drives. In addition to the timed disk access test, the hard disk test also reports on disk capacity and issues warnings when a disk nears its capacity.

This portion of the System Diagnosis module takes two parameters. The parameters, hard_disk_test and hd_count, function almost identically to the similarly named parameters of the memory test. The hard_disk_test parameter determines whether the hard disk test will be run and the hd_count parameter indicates the number of times the hard disk test will execute its test loop. The following table lists all of the parameters for the System Diagnosis module.

Parameter	Values	Description
execute	yes / no	Whether to execute the module
mem_test	yes / no	Whether to execute the memory test
hd_test	yes / no	Whether to execute the hard disk test
cpu_test	yes / no	Whether to execute the cpu test
mem_count	integer > 0	A positive integer that determines how many times the memory test will execute
hd_count	integer > 0	A positive integer that determines how many times the memory test will execute

3.1.4. Performance Module

The Performance module's task is to test and report on the performance capabilities of a cluster's network and compute nodes. This module contains two performance evaluation tools. The first, called Linpack, is a linear algebra package and is used by Bright to test the speed of individual compute nodes [8]. The second is called Netpipe. Netpipe is a network protocol independent performance evaluation tool used by Bright to gauge the performance of cluster networks [5]. To execute these tests, you must set their respective parameters in the Performance section of the parameter file to “yes”. Both tests are executed by default but can be easily turned off, just as for any other test or module, by setting the parameters to “no”.

In addition to the parameters that determine whether the tests will run or not, each of these two tests have their own set of parameters. The Linpack test takes two values from the parameter file, precision and rolled. The precision parameter needs to be set to either “double” or “single” that, of course, determines the precision on which the Linpack benchmark will be

based. The rolled parameter influences the way the Linpack utility is run by changing the way its program loops are executed. If rolled is set to “off” then unrolled versions of some program loops are used. This means that some sequences of instructions are not executed within loops. If rolled is set to “on” then these instructions are executed within loops. Unrolled versions of code are used to avoid the overhead of loop management. In more recent machines and compilers however, there is a higher performance penalty associated with unrolled code than there is with rolled code. This is why the default value of the rolled parameter is 'on' and most users will want to leave it that way.

Currently Netpipe has only one parameter associated with it, the net_count parameter. The value of this parameter determines the number of Netpipe processes that are started on each node in a cluster. One important thing to note here is that the larger the value of net_count is, the greater the load placed on the network. It is a good idea to start out with a low value for net_count and gradually increase it as you use the utility in order to determine just how much your network can handle. By default net_count is set to one. The following table lists all of the parameters for the Performance module.

Parameter	Values	Description
execute	yes / no	Whether to execute the module
linpack	yes / no	Whether to execute Linpack
netpipe	yes / no	Whether to execute Netpipe
precision	single / double	Determines whether Linpack will use double or single precision floating point values in its calculations
rolled	yes / no	Determines whether Linpack will use rolled or unrolled precision in it calculations
net_count	integer > 0	A positive integer that determines how many instances of netpipe will be

4. Using Bright

Once you have Bright configured, i.e., by modifying the beowulf.conf file to your needs, (see the installation and Configuration section of this manual) you are ready to use the utility. This section describes Bright's usage.

4.1. Running Bright

Before running the Bright Utility, you need to prepare the configuration file (section 2.3.1) and the parameter file (section 2.3.3). Before you do this you should make sure you have read section 3 of this manual called The Modules.

To run Bright all you need to do is to execute the script /home/<user>/bright-0.99/modules/main/main.pl. To do this just switch to the directory where the main.pl script is located and type in "./main.pl" at the prompt. If it is not already present, the utility will create the directory /home/<user>/.bright where it will store temporary files as the utility is running. Bright will then go on to parse the parameter file. Any errors within the parameter file will be reported before the utility is run. Once this step is complete Bright will execute whichever modules are set to run, including user-defined tests.

In most cases, depending on which modules are run and how they are configured, the utility could take quite a while to complete. For example, we tested the utility on a 16 node cluster will all of the tests enable and it took 35 minutes to complete. Hence, you may find that Bright is best utilized when run overnight as a cron job, or when cluster is not being used heavily.

During execution Bright creates and stores its logs in /home/<user>/bright-0.99/log. The log filenames are based on the date and time so that records of past executions can be kept easily. Once Bright has finished, the user will be left with two new files in Bright's log directory

(/home/<user>/bright-0.99/log), one called <date/time>.log and <date/time>.sum, where <date/time> is the date and time of that execution of Bright. The file with the .log suffix is a fairly verbose and lengthy report of everything that went on when the utility was run. The .sum file is a summarized version of the .log file. It contains more general information, such as which nodes are reachable, what errors occurred, if any, which packages are missing from which nodes, etc.

4.2. Interpreting the Results

This section will describe the summary file and the log file that are generated by Bright. Figure 6 shows an example summary file that was generated when Bright was run on one of the clusters at Goddard Space Flight Center.

```
Bright execution Mon Dec 13 19:25:43 1999 on host hrothgar
Summary:
------------
Completeness Module:
  The following packages were not installed:
   pvm

Compatibility Module:
    Node Status:
        Local network interface is up
        h1 is down
        h10 is up
        h11 is up
        h12 is up
        h13 is up
        h14 is up
        h15 is up
        h16 is up
        h2 is up
        h3 is up
        h4 is up
        h5 is up
        h6 is up
        h7 is up
        h8 is up
        h9 is up
    Packages Installed:
        pvm: not found on any node
        ssh:  h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16

System Diagnosis Module:

  Disk test times:
    Minimum = 26770000 microsecs on node h15
    Maximum = 55310000 microsecs on node h16
    Average = 31671333.3333333 microsecs

  CPU Information:
    Worst precision = 17 decimal places on node h8.
    Best precision = 17 decimal places on node h8.
    Average precision = 17 decimal places.

Performance Module:
  Linpack Results:

    Minimum = 13.04 Mflops on node h12
    Maximum = 13.53 Mflops on node h11
    Average = 13.324 Mflops
```

Figure 6: Example summary file

As you can see from Figure 6, each module has a section in the summary file where key information about the module's execution is stored. In addition to system information, each section will also contain a listing of any errors that occurred during that module's execution.

Even though the different sections of the summary file are virtually self-explanatory, there is one issue you need to be aware of before interpreting some of the results of the tests. It is sometimes a bad idea for the user to base cluster observations purely on the summary file. An example of this is disk test timing. In Figure 6, the minimum and maximum disk test times are very different. At first glance one would think that something might be wrong with node h16, the node with a test time of nearly twice the average. This is in fact not the case. The greater time h16 took to complete was expected because h16 has twice as much memory as the other nodes. This means that the size of the chunks of disk space used in the hard disk test were each twice as large on h16 as the other nodes, causing the hard disk test to take about twice a long as all the other nodes. If someone looking at the summary file did not already know that node h16 had more memory, or if he or she had not looked at the log file to see exactly how much memory h16 had, they may have jumped to an incorrect conclusion.

The log file that Bright outputs is not nearly as easy to analyze as the summary file. Figure 7 is an example of an entry in the log file.

```
bright: performance: linpack: h13: Module started Mon Dec 13 13:47:10 EST 1999
bright: h13: performance: linpack: LINPACK BENCHMARK
bright: h13: performance: linpack: CPU
bright: h13: performance: linpack: Clock MHz
bright: h13: performance: linpack: Cache
bright: h13: performance: linpack: Rolling          Unrolled
bright: h13: performance: linpack: Precision        Single
bright: h13: performance: linpack: norm. resid          1.9
bright: h13: performance: linpack: resid            4.52336171e-05
bright: h13: performance: linpack: machep           1.19209290e-07
bright: h13: performance: linpack: x[0]-1           -1.31130219e-05
bright: h13: performance: linpack: x[n-1]-1         -1.30534172e-05
bright: h13: performance: linpack: smatgen 1 seconds        0.01087
bright: h13: performance: linpack: smatgen 2 seconds        0.01087
bright: h13: performance: linpack: Repetitions              82
bright: h13: performance: linpack: Leading dimension        201
bright: h13: performance: linpack:                  sdgefa    sdgesl    total    Mflops
bright: h13: performance: linpack: 1 pass seconds       0.05000   0.00000   0.05000
bright: h13: performance: linpack: Repeat seconds       0.04998   0.00146   0.05144    13.35
bright: h13: performance: linpack: Repeat seconds       0.04998   0.00159   0.05156    13.32
bright: h13: performance: linpack: Repeat seconds       0.04998   0.00146   0.05144    13.35
bright: h13: performance: linpack: Repeat seconds       0.04998   0.00159   0.05156    13.32
bright: h13: performance: linpack: Repeat seconds       0.04998   0.00146   0.05144    13.35
bright: h13: performance: linpack: Average                                    13.34
bright: h13: performance: linpack: Leading dimension        200
bright: h13: performance: linpack: Repeat seconds       0.05022   0.00146   0.05169    13.29
bright: h13: performance: linpack: Repeat seconds       0.05034   0.00146   0.05181    13.25
bright: h13: performance: linpack: Repeat seconds       0.05022   0.00159   0.05181    13.25
bright: h13: performance: linpack: Repeat seconds       0.05022   0.00159   0.05181    13.25
bright: h13: performance: linpack: Repeat seconds       0.05022   0.00159   0.05181    13.25
bright: h13: performance: linpack: Average                                    13.26
bright: h13: performance: linpack: Finished test on Mon Dec 13 13:48:42 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h13
```

Figure 7: Example log file entry

Every entry in the log file has the same form. Each line begins with the phrase "bright: ". This is followed by hostname of the node that is executing the test followed by a colon. Then the module name followed by a colon is printed. Finally, the name of the test within the module follwed by a colon is printed. After all of the beginning information is printed, the actual output from the test will be printed. We tried to make it as easy as possible to search through the log file in order to find the results from an individual test. The summary file is much more compact and is easier to read.

5. Conclusions

We hope this manual has been helpful and has covered all the information you will need to know to get Bright up and running on your own cluster. As was noted throughout this document however, this manual describes the operation of the project version of Bright. As time goes on we hope to add more functionality, improve our reporting mechanisms and, hopefully, officially release it under the Gnu General Public License.

5.1. Contact Information

Group Member	**Email address**
Dennis Mattoon	dmattoon@wpi.edu
Jeff Cilley	jcilley@wpi.edu
Aaron Chandler-Worth	mandpoe@wpi.edu
Project Advisors	
Professor David Finkel	dfinkel@wpi.edu
Professor David Brown	dcb@wpi.edu

We hope to eventually have a web site that will contain the most up to date information about our utility including the most recent version.

Appendix B: An Example Log File

```
bright: Execution began Mon Dec 13 19:25:43 1999 on host hrothgar
bright: Initializing temporary directory /home/dmattoon/.bright
bright: Verifying presence of config file
bright: Config file found (**path to config file**)
bright: Opening parameter file (**path to parameter file**)
bright: Attempting to parse parameter file
bright: Parsing complete, running modules
bright: Beginning completeness module
bright: completeness: Module started on hrothgar Mon Dec 13 19:25:43 1999
bright: completeness: Temporary file bright-compl.hrothgar created in /home/dmattoon/.bright
bright: completeness: Looking for path to rpm command
bright: completeness: Command found, path: /bin/rpm
bright: completeness: Parsing config file for package list
bright: completeness: Package list found
bright: completeness: Checking for pvm package
bright: completeness: Warning: pvm is not installed
bright: completeness: Checking for ssh package
bright: completeness: ssh is installed, packages: ssh-1.2.26-2
bright: completeness: Module finished Mon Dec 13 19:25:43 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compl.hrothgar
bright: Executing completeness-nested user-defined tests (if any)
bright: Gathering node hostnames from config file
bright: Finished gathering hostnames
bright: Checking status of local network interface
bright: Local network interface is up
bright: Determining reachability of each node (ping)
bright: Node status:
bright: h1 is down
bright: h2 is up
bright: h3 is up
bright: h4 is up
bright: h5 is up
bright: h6 is up
bright: h7 is up
bright: h8 is up
bright: h9 is up
bright: h10 is up
bright: h11 is up
bright: h12 is up
bright: h13 is up
bright: h14 is up
bright: h15 is up
bright: h16 is up
bright: Finished pinging nodes
bright: Beginning compatibility module on each reachable node
bright: compatibility: h2: Module started on Mon Dec 13 17:40:32 1999
bright: compatibility: h2: Temporary file bright-compat.h2 created in /home/dmattoon/.bright
bright: compatibility: h2: Deleting temporary file /home/dmattoon/.bright/bright-compl.h2
bright: compatibility: h2: Temporary file bright-compl.h2 created in /home/dmattoon/.bright
bright: compatibility: h2: Looking for path to rpm command
bright: compatibility: h2: Command found, path: /bin/rpm
bright: compatibility: h2: Parsing config file for package list
bright: compatibility: h2: Package list found
```

bright: compatibility: h2: Checking for pvm package
bright: compatibility: h2: Warning: pvm is not installed
bright: compatibility: h2: Checking for ssh package
bright: compatibility: h2: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h2: Module finished Mon Dec 13 17:40:36 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h2
bright: compatibility: h3: Module started on Mon Dec 13 12:22:15 1999
bright: compatibility: h3: Temporary file bright-compat.h3 created in /home/dmattoon/.bright
bright: compatibility: h3: Deleting temporary file /home/dmattoon/.bright/bright-compl.h3
bright: compatibility: h3: Temporary file bright-compl.h3 created in /home/dmattoon/.bright
bright: compatibility: h3: Looking for path to rpm command
bright: compatibility: h3: Command found, path: /bin/rpm
bright: compatibility: h3: Parsing config file for package list
bright: compatibility: h3: Package list found
bright: compatibility: h3: Checking for pvm package
bright: compatibility: h3: Warning: pvm is not installed
bright: compatibility: h3: Checking for ssh package
bright: compatibility: h3: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h3: Module finished Mon Dec 13 12:22:16 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h3
bright: compatibility: h4: Module started on Mon Dec 13 12:06:32 1999
bright: compatibility: h4: Temporary file bright-compat.h4 created in /home/dmattoon/.bright
bright: compatibility: h4: Deleting temporary file /home/dmattoon/.bright/bright-compl.h4
bright: compatibility: h4: Temporary file bright-compl.h4 created in /home/dmattoon/.bright
bright: compatibility: h4: Looking for path to rpm command
bright: compatibility: h4: Command found, path: /bin/rpm
bright: compatibility: h4: Parsing config file for package list
bright: compatibility: h4: Package list found
bright: compatibility: h4: Checking for pvm package
bright: compatibility: h4: Warning: pvm is not installed
bright: compatibility: h4: Checking for ssh package
bright: compatibility: h4: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h4: Module finished Mon Dec 13 12:06:33 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h4
bright: compatibility: h5: Module started on Mon Dec 13 19:07:58 1999
bright: compatibility: h5: Temporary file bright-compat.h5 created in /home/dmattoon/.bright
bright: compatibility: h5: Deleting temporary file /home/dmattoon/.bright/bright-compl.h5
bright: compatibility: h5: Temporary file bright-compl.h5 created in /home/dmattoon/.bright
bright: compatibility: h5: Looking for path to rpm command
bright: compatibility: h5: Command found, path: /bin/rpm
bright: compatibility: h5: Parsing config file for package list
bright: compatibility: h5: Package list found
bright: compatibility: h5: Checking for pvm package
bright: compatibility: h5: Warning: pvm is not installed
bright: compatibility: h5: Checking for ssh package
bright: compatibility: h5: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h5: Module finished Mon Dec 13 19:07:59 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h5
bright: compatibility: h6: Module started on Mon Dec 13 12:23:37 1999
bright: compatibility: h6: Temporary file bright-compat.h6 created in /home/dmattoon/.bright
bright: compatibility: h6: Deleting temporary file /home/dmattoon/.bright/bright-compl.h6
bright: compatibility: h6: Temporary file bright-compl.h6 created in /home/dmattoon/.bright
bright: compatibility: h6: Looking for path to rpm command
bright: compatibility: h6: Command found, path: /bin/rpm
bright: compatibility: h6: Parsing config file for package list
bright: compatibility: h6: Package list found

bright: compatibility: h6: Checking for pvm package
bright: compatibility: h6: Warning: pvm is not installed
bright: compatibility: h6: Checking for ssh package
bright: compatibility: h6: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h6: Module finished Mon Dec 13 12:23:38 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h6
bright: compatibility: h7: Module started on Mon Dec 13 12:18:58 1999
bright: compatibility: h7: Temporary file bright-compat.h7 created in /home/dmattoon/.bright
bright: compatibility: h7: Deleting temporary file /home/dmattoon/.bright/bright-compl.h7
bright: compatibility: h7: Temporary file bright-compl.h7 created in /home/dmattoon/.bright
bright: compatibility: h7: Looking for path to rpm command
bright: compatibility: h7: Command found, path: /bin/rpm
bright: compatibility: h7: Parsing config file for package list
bright: compatibility: h7: Package list found
bright: compatibility: h7: Checking for pvm package
bright: compatibility: h7: Warning: pvm is not installed
bright: compatibility: h7: Checking for ssh package
bright: compatibility: h7: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h7: Module finished Mon Dec 13 12:18:59 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h7
bright: compatibility: h8: Module started on Tue Dec 14 12:18:52 1999
bright: compatibility: h8: Temporary file bright-compat.h8 created in /home/dmattoon/.bright
bright: compatibility: h8: Deleting temporary file /home/dmattoon/.bright/bright-compl.h8
bright: compatibility: h8: Temporary file bright-compl.h8 created in /home/dmattoon/.bright
bright: compatibility: h8: Looking for path to rpm command
bright: compatibility: h8: Command found, path: /bin/rpm
bright: compatibility: h8: Parsing config file for package list
bright: compatibility: h8: Package list found
bright: compatibility: h8: Checking for pvm package
bright: compatibility: h8: Warning: pvm is not installed
bright: compatibility: h8: Checking for ssh package
bright: compatibility: h8: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h8: Module finished Tue Dec 14 12:18:52 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h8
bright: compatibility: h9: Module started on Tue Dec 14 00:52:55 1999
bright: compatibility: h9: Temporary file bright-compat.h9 created in /home/dmattoon/.bright
bright: compatibility: h9: Deleting temporary file /home/dmattoon/.bright/bright-compl.h9
bright: compatibility: h9: Temporary file bright-compl.h9 created in /home/dmattoon/.bright
bright: compatibility: h9: Looking for path to rpm command
bright: compatibility: h9: Command found, path: /bin/rpm
bright: compatibility: h9: Parsing config file for package list
bright: compatibility: h9: Package list found
bright: compatibility: h9: Checking for pvm package
bright: compatibility: h9: Warning: pvm is not installed
bright: compatibility: h9: Checking for ssh package
bright: compatibility: h9: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h9: Module finished Tue Dec 14 00:52:56 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h9
bright: compatibility: h10: Module started on Mon Dec 13 12:01:47 1999
bright: compatibility: h10: Temporary file bright-compat.h10 created in /home/dmattoon/.bright
bright: compatibility: h10: Deleting temporary file /home/dmattoon/.bright/bright-compl.h10
bright: compatibility: h10: Temporary file bright-compl.h10 created in /home/dmattoon/.bright
bright: compatibility: h10: Looking for path to rpm command
bright: compatibility: h10: Command found, path: /bin/rpm
bright: compatibility: h10: Parsing config file for package list
bright: compatibility: h10: Package list found

bright: compatibility: h10: Checking for pvm package
bright: compatibility: h10: Warning: pvm is not installed
bright: compatibility: h10: Checking for ssh package
bright: compatibility: h10: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h10: Module finished Mon Dec 13 12:01:48 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h10
bright: compatibility: h11: Module started on Mon Dec 13 12:48:19 1999
bright: compatibility: h11: Temporary file bright-compat.h11 created in /home/dmattoon/.bright
bright: compatibility: h11: Deleting temporary file /home/dmattoon/.bright/bright-compl.h11
bright: compatibility: h11: Temporary file bright-compl.h11 created in /home/dmattoon/.bright
bright: compatibility: h11: Looking for path to rpm command
bright: compatibility: h11: Command found, path: /bin/rpm
bright: compatibility: h11: Parsing config file for package list
bright: compatibility: h11: Package list found
bright: compatibility: h11: Checking for pvm package
bright: compatibility: h11: Warning: pvm is not installed
bright: compatibility: h11: Checking for ssh package
bright: compatibility: h11: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h11: Module finished Mon Dec 13 12:48:20 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h11
bright: compatibility: h12: Module started on Mon Dec 13 12:09:23 1999
bright: compatibility: h12: Temporary file bright-compat.h12 created in /home/dmattoon/.bright
bright: compatibility: h12: Deleting temporary file /home/dmattoon/.bright/bright-compl.h12
bright: compatibility: h12: Temporary file bright-compl.h12 created in /home/dmattoon/.bright
bright: compatibility: h12: Looking for path to rpm command
bright: compatibility: h12: Command found, path: /bin/rpm
bright: compatibility: h12: Parsing config file for package list
bright: compatibility: h12: Package list found
bright: compatibility: h12: Checking for pvm package
bright: compatibility: h12: Warning: pvm is not installed
bright: compatibility: h12: Checking for ssh package
bright: compatibility: h12: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h12: Module finished Mon Dec 13 12:09:24 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h12
bright: compatibility: h13: Module started on Mon Dec 13 13:18:34 1999
bright: compatibility: h13: Temporary file bright-compat.h13 created in /home/dmattoon/.bright
bright: compatibility: h13: Deleting temporary file /home/dmattoon/.bright/bright-compl.h13
bright: compatibility: h13: Temporary file bright-compl.h13 created in /home/dmattoon/.bright
bright: compatibility: h13: Looking for path to rpm command
bright: compatibility: h13: Command found, path: /bin/rpm
bright: compatibility: h13: Parsing config file for package list
bright: compatibility: h13: Package list found
bright: compatibility: h13: Checking for pvm package
bright: compatibility: h13: Warning: pvm is not installed
bright: compatibility: h13: Checking for ssh package
bright: compatibility: h13: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h13: Module finished Mon Dec 13 13:18:35 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h13
bright: compatibility: h14: Module started on Mon Dec 13 13:07:02 1999
bright: compatibility: h14: Temporary file bright-compat.h14 created in /home/dmattoon/.bright
bright: compatibility: h14: Deleting temporary file /home/dmattoon/.bright/bright-compl.h14
bright: compatibility: h14: Temporary file bright-compl.h14 created in /home/dmattoon/.bright
bright: compatibility: h14: Looking for path to rpm command
bright: compatibility: h14: Command found, path: /bin/rpm
bright: compatibility: h14: Parsing config file for package list
bright: compatibility: h14: Package list found

bright: compatibility: h14: Checking for pvm package
bright: compatibility: h14: Warning: pvm is not installed
bright: compatibility: h14: Checking for ssh package
bright: compatibility: h14: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h14: Module finished Mon Dec 13 13:07:03 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h14
bright: compatibility: h15: Module started on Mon Dec 13 12:11:25 1999
bright: compatibility: h15: Temporary file bright-compat.h15 created in /home/dmattoon/.bright
bright: compatibility: h15: Deleting temporary file /home/dmattoon/.bright/bright-compl.h15
bright: compatibility: h15: Temporary file bright-compl.h15 created in /home/dmattoon/.bright
bright: compatibility: h15: Looking for path to rpm command
bright: compatibility: h15: Command found, path: /bin/rpm
bright: compatibility: h15: Parsing config file for package list
bright: compatibility: h15: Package list found
bright: compatibility: h15: Checking for pvm package
bright: compatibility: h15: Warning: pvm is not installed
bright: compatibility: h15: Checking for ssh package
bright: compatibility: h15: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h15: Module finished Mon Dec 13 12:11:26 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h15
bright: compatibility: h16: Module started on Mon Dec 13 12:32:43 1999
bright: compatibility: h16: Temporary file bright-compat.h16 created in /home/dmattoon/.bright
bright: compatibility: h16: Deleting temporary file /home/dmattoon/.bright/bright-compl.h16
bright: compatibility: h16: Temporary file bright-compl.h16 created in /home/dmattoon/.bright
bright: compatibility: h16: Looking for path to rpm command
bright: compatibility: h16: Command found, path: /bin/rpm
bright: compatibility: h16: Parsing config file for package list
bright: compatibility: h16: Package list found
bright: compatibility: h16: Checking for pvm package
bright: compatibility: h16: Warning: pvm is not installed
bright: compatibility: h16: Checking for ssh package
bright: compatibility: h16: ssh is installed, packages: ssh-1.2.26-2
bright: compatibility: h16: Module finished Mon Dec 13 12:32:44 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-compat.h16
bright: Executing compatibility-nested user-defined tests (if any)
bright: Beginning system diagnosis module on hrothgar
bright: sysdiag: PATH: /home/dmattoon/.bright/bright-sysdiag.hrothgar
bright: sysdiag: The system diagnosis module has completed on hrothgar
bright: sysdiag: Now executing system diagnosis module on the nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h2
bright: sysdiag: h2: Module started
bright: sysdiag: h2: Disk Information:
bright: sysdiag: h2: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h2: /dev/hda1 1.1G 233M 830M 22% /
bright: sysdiag: h2: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h2: Memory Information:
bright: sysdiag: h2: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h2: 31563776 28971008 2592768 4968448 5963776 17047552
bright: sysdiag: h2: Beginning memory testing
bright: sysdiag: h2: Allocating 2592768 bytes
bright: sysdiag: h2: Memory test completed successfully!
bright: sysdiag: h2: Beginning cpu testing
bright: sysdiag: h2: Accurate to 20 digits in calculations
bright: sysdiag: h2: Accurate to 20 digits in memory
bright: sysdiag: h2: Precise to 17 decimal places
bright: sysdiag: h2: Beginning hard disk testing

bright: sysdiag: h2: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h2: hard disk test took 29110000msec to complete
bright: sysdiag: h2: Hard disk test completed successfully!
bright: sysdiag: h2: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h3
bright: sysdiag: h3: Module started
bright: sysdiag: h3: Disk Information:
bright: sysdiag: h3: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h3: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h3: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h3: Memory Information:
bright: sysdiag: h3: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h3: 31563776 28332032 3231744 3649536 8445952 15327232
bright: sysdiag: h3: Beginning memory testing
bright: sysdiag: h3: Allocating 3231744 bytes
bright: sysdiag: h3: Memory test completed successfully!
bright: sysdiag: h3: Beginning cpu testing
bright: sysdiag: h3: Accurate to 20 digits in calculations
bright: sysdiag: h3: Accurate to 20 digits in memory
bright: sysdiag: h3: Precise to 17 decimal places
bright: sysdiag: h3: Beginning hard disk testing
bright: sysdiag: h3: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h3: hard disk test took 29660000msec to complete
bright: sysdiag: h3: Hard disk test completed successfully!
bright: sysdiag: h3: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h4
bright: sysdiag: h4: Module started
bright: sysdiag: h4: Disk Information:
bright: sysdiag: h4: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h4: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h4: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h4: Memory Information:
bright: sysdiag: h4: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h4: 31563776 28217344 3346432 6373376 7479296 15884288
bright: sysdiag: h4: Beginning memory testing
bright: sysdiag: h4: Allocating 3346432 bytes
bright: sysdiag: h4: Memory test completed successfully!
bright: sysdiag: h4: Beginning cpu testing
bright: sysdiag: h4: Accurate to 20 digits in calculations
bright: sysdiag: h4: Accurate to 20 digits in memory
bright: sysdiag: h4: Precise to 17 decimal places
bright: sysdiag: h4: Beginning hard disk testing
bright: sysdiag: h4: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h4: hard disk test took 29650000msec to complete
bright: sysdiag: h4: Hard disk test completed successfully!
bright: sysdiag: h4: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h5
bright: sysdiag: h5: Module started
bright: sysdiag: h5: Disk Information:
bright: sysdiag: h5: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h5: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h5: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h5: Memory Information:

bright: sysdiag: h5: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h5: 31563776 28958720 2605056 6352896 7737344 16355328
bright: sysdiag: h5: Beginning memory testing
bright: sysdiag: h5: Allocating 2605056 bytes
bright: sysdiag: h5: Memory test completed successfully!
bright: sysdiag: h5: Beginning cpu testing
bright: sysdiag: h5: Accurate to 20 digits in calculations
bright: sysdiag: h5: Accurate to 20 digits in memory
bright: sysdiag: h5: Precise to 17 decimal places
bright: sysdiag: h5: Beginning hard disk testing
bright: sysdiag: h5: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h5: hard disk test took 28990000msec to complete
bright: sysdiag: h5: Hard disk test completed successfully!
bright: sysdiag: h5: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h6
bright: sysdiag: h6: Module started
bright: sysdiag: h6: Disk Information:
bright: sysdiag: h6: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h6: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h6: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h6: Memory Information:
bright: sysdiag: h6: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h6: 31563776 27910144 3653632 3424256 8040448 15314944
bright: sysdiag: h6: Beginning memory testing
bright: sysdiag: h6: Allocating 3653632 bytes
bright: sysdiag: h6: Memory test completed successfully!
bright: sysdiag: h6: Beginning cpu testing
bright: sysdiag: h6: Accurate to 20 digits in calculations
bright: sysdiag: h6: Accurate to 20 digits in memory
bright: sysdiag: h6: Precise to 17 decimal places
bright: sysdiag: h6: Beginning hard disk testing
bright: sysdiag: h6: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h6: hard disk test took 30390000msec to complete
bright: sysdiag: h6: Hard disk test completed successfully!
bright: sysdiag: h6: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h7
bright: sysdiag: h7: Module started
bright: sysdiag: h7: Disk Information:
bright: sysdiag: h7: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h7: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h7: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h7: Memory Information:
bright: sysdiag: h7: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h7: 31563776 27955200 3608576 3428352 8171520 15355904
bright: sysdiag: h7: Beginning memory testing
bright: sysdiag: h7: Allocating 3608576 bytes
bright: sysdiag: h7: Memory test completed successfully!
bright: sysdiag: h7: Beginning cpu testing
bright: sysdiag: h7: Accurate to 20 digits in calculations
bright: sysdiag: h7: Accurate to 20 digits in memory
bright: sysdiag: h7: Precise to 17 decimal places
bright: sysdiag: h7: Beginning hard disk testing
bright: sysdiag: h7: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h7: hard disk test took 29650000msec to complete

bright: sysdiag: h7: Hard disk test completed successfully!
bright: sysdiag: h7: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h8
bright: sysdiag: h8: Module started
bright: sysdiag: h8: Disk Information:
bright: sysdiag: h8: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h8: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h8: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h8: Memory Information:
bright: sysdiag: h8: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h8: 31563776 28962816 2600960 3424256 8912896 15495168
bright: sysdiag: h8: Beginning memory testing
bright: sysdiag: h8: Allocating 2600960 bytes
bright: sysdiag: h8: Memory test completed successfully!
bright: sysdiag: h8: Beginning cpu testing
bright: sysdiag: h8: Accurate to 20 digits in calculations
bright: sysdiag: h8: Accurate to 20 digits in memory
bright: sysdiag: h8: Precise to 17 decimal places
bright: sysdiag: h8: Beginning hard disk testing
bright: sysdiag: h8: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h8: hard disk test took 29590000msec to complete
bright: sysdiag: h8: Hard disk test completed successfully!
bright: sysdiag: h8: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h9
bright: sysdiag: h9: Module started
bright: sysdiag: h9: Disk Information:
bright: sysdiag: h9: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h9: /dev/hda1 7.5G 232M 6.9G 3% /
bright: sysdiag: h9: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h9: Memory Information:
bright: sysdiag: h9: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h9: 31563776 28954624 2609152 3432448 8699904 15704064
bright: sysdiag: h9: Beginning memory testing
bright: sysdiag: h9: Allocating 2609152 bytes
bright: sysdiag: h9: Memory test completed successfully!
bright: sysdiag: h9: Beginning cpu testing
bright: sysdiag: h9: Accurate to 20 digits in calculations
bright: sysdiag: h9: Accurate to 20 digits in memory
bright: sysdiag: h9: Precise to 17 decimal places
bright: sysdiag: h9: Beginning hard disk testing
bright: sysdiag: h9: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h9: hard disk test took 36370000msec to complete
bright: sysdiag: h9: Hard disk test completed successfully!
bright: sysdiag: h9: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h10
bright: sysdiag: h10: Module started
bright: sysdiag: h10: Disk Information:
bright: sysdiag: h10: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h10: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h10: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h10: Memory Information:
bright: sysdiag: h10: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h10: 31563776 27815936 3747840 6234112 8101888 14843904

bright: sysdiag: h10: Beginning memory testing
bright: sysdiag: h10: Allocating 3747840 bytes
bright: sysdiag: h10: Memory test completed successfully!
bright: sysdiag: h10: Beginning cpu testing
bright: sysdiag: h10: Accurate to 20 digits in calculations
bright: sysdiag: h10: Accurate to 20 digits in memory
bright: sysdiag: h10: Precise to 17 decimal places
bright: sysdiag: h10: Beginning hard disk testing
bright: sysdiag: h10: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h10: hard disk test took 29640000msec to complete
bright: sysdiag: h10: Hard disk test completed successfully!
bright: sysdiag: h10: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h11
bright: sysdiag: h11: Module started
bright: sysdiag: h11: Disk Information:
bright: sysdiag: h11: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h11: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h11: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h11: Memory Information:
bright: sysdiag: h11: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h11: 31563776 28045312 3518464 3432448 8212480 15257600
bright: sysdiag: h11: Beginning memory testing
bright: sysdiag: h11: Allocating 3518464 bytes
bright: sysdiag: h11: Memory test completed successfully!
bright: sysdiag: h11: Beginning cpu testing
bright: sysdiag: h11: Accurate to 20 digits in calculations
bright: sysdiag: h11: Accurate to 20 digits in memory
bright: sysdiag: h11: Precise to 17 decimal places
bright: sysdiag: h11: Beginning hard disk testing
bright: sysdiag: h11: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h11: hard disk test took 29660000msec to complete
bright: sysdiag: h11: Hard disk test completed successfully!
bright: sysdiag: h11: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h12
bright: sysdiag: h12: Module started
bright: sysdiag: h12: Disk Information:
bright: sysdiag: h12: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h12: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h12: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h12: Memory Information:
bright: sysdiag: h12: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h12: 31563776 27828224 3735552 3641344 7913472 15347712
bright: sysdiag: h12: Beginning memory testing
bright: sysdiag: h12: Allocating 3735552 bytes
bright: sysdiag: h12: Memory test completed successfully!
bright: sysdiag: h12: Beginning cpu testing
bright: sysdiag: h12: Accurate to 20 digits in calculations
bright: sysdiag: h12: Accurate to 20 digits in memory
bright: sysdiag: h12: Precise to 17 decimal places
bright: sysdiag: h12: Beginning hard disk testing
bright: sysdiag: h12: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h12: hard disk test took 30270000msec to complete
bright: sysdiag: h12: Hard disk test completed successfully!
bright: sysdiag: h12: Module finished

bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h13
bright: sysdiag: h13: Module started
bright: sysdiag: h13: Disk Information:
bright: sysdiag: h13: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h13: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h13: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h13: Memory Information:
bright: sysdiag: h13: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h13: 31563776 27979776 3584000 6262784 7806976 15335424
bright: sysdiag: h13: Beginning memory testing
bright: sysdiag: h13: Allocating 3584000 bytes
bright: sysdiag: h13: Memory test completed successfully!
bright: sysdiag: h13: Beginning cpu testing
bright: sysdiag: h13: Accurate to 20 digits in calculations
bright: sysdiag: h13: Accurate to 20 digits in memory
bright: sysdiag: h13: Precise to 17 decimal places
bright: sysdiag: h13: Beginning hard disk testing
bright: sysdiag: h13: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h13: hard disk test took 29690000msec to complete
bright: sysdiag: h13: Hard disk test completed successfully!
bright: sysdiag: h13: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h14
bright: sysdiag: h14: Module started
bright: sysdiag: h14: Disk Information:
bright: sysdiag: h14: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h14: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h14: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h14: Memory Information:
bright: sysdiag: h14: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h14: 31563776 28860416 2703360 6361088 7847936 16175104
bright: sysdiag: h14: Beginning memory testing
bright: sysdiag: h14: Allocating 2703360 bytes
bright: sysdiag: h14: Memory test completed successfully!
bright: sysdiag: h14: Beginning cpu testing
bright: sysdiag: h14: Accurate to 20 digits in calculations
bright: sysdiag: h14: Accurate to 20 digits in memory
bright: sysdiag: h14: Precise to 17 decimal places
bright: sysdiag: h14: Beginning hard disk testing
bright: sysdiag: h14: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h14: hard disk test took 30320000msec to complete
bright: sysdiag: h14: Hard disk test completed successfully!
bright: sysdiag: h14: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h15
bright: sysdiag: h15: Module started
bright: sysdiag: h15: Disk Information:
bright: sysdiag: h15: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h15: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h15: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h15: Memory Information:
bright: sysdiag: h15: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h15: 31563776 27885568 3678208 6246400 8310784 14770176
bright: sysdiag: h15: Beginning memory testing
bright: sysdiag: h15: Allocating 3678208 bytes

bright: sysdiag: h15: Memory test completed successfully!
bright: sysdiag: h15: Beginning cpu testing
bright: sysdiag: h15: Accurate to 20 digits in calculations
bright: sysdiag: h15: Accurate to 20 digits in memory
bright: sysdiag: h15: Precise to 17 decimal places
bright: sysdiag: h15: Beginning hard disk testing
bright: sysdiag: h15: allocating 15781888, 15781888, and 15781888 bytes
bright: sysdiag: h15: hard disk test took 26770000msec to complete
bright: sysdiag: h15: Hard disk test completed successfully!
bright: sysdiag: h15: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: sysdiag: rsh output: PATH: /home/dmattoon/.bright/bright-sysdiag.h16
bright: sysdiag: h16: Module started
bright: sysdiag: h16: Disk Information:
bright: sysdiag: h16: Filesystem Size Used Avail Capacity Mounted on
bright: sysdiag: h16: /dev/hda1 1.1G 232M 830M 22% /
bright: sysdiag: h16: 192.168.1.17:/home 3.7G 2.3G 1.2G 66% /home
bright: sysdiag: h16: Memory Information:
bright: sysdiag: h16: Total: Used: Free: Shared: Buffers: Cached:
bright: sysdiag: h16: 64655360 59572224 5083136 6393856 35766272 15384576
bright: sysdiag: h16: Beginning memory testing
bright: sysdiag: h16: Allocating 5083136 bytes
bright: sysdiag: h16: Memory test completed successfully!
bright: sysdiag: h16: Beginning cpu testing
bright: sysdiag: h16: Accurate to 20 digits in calculations
bright: sysdiag: h16: Accurate to 20 digits in memory
bright: sysdiag: h16: Precise to 17 decimal places
bright: sysdiag: h16: Beginning hard disk testing
bright: sysdiag: h16: allocating 32327680, 32327680, and 32327680 bytes
bright: sysdiag: h16: hard disk test took 55310000msec to complete
bright: sysdiag: h16: Hard disk test completed successfully!
bright: sysdiag: h16: Module finished
bright: sysdiag: Deleteing temporary files created by nodes
bright: performance: linpack: Starting Linpack test.
bright: performance: linpack: h7: Module started Mon Dec 13 12:38:38 EST 1999
bright: h7: performance: linpack: LINPACK BENCHMARK
bright: h7: performance: linpack: CPU
bright: h7: performance: linpack: Clock MHz
bright: h7: performance: linpack: Cache
bright: h7: performance: linpack: Rolling Unrolled
bright: h7: performance: linpack: Precision Single
bright: h7: performance: linpack: norm. resid 1.9
bright: h7: performance: linpack: resid 4.52336171e-05
bright: h7: performance: linpack: machep 1.19209290e-07
bright: h7: performance: linpack: x[0]-1 -1.31130219e-05
bright: h7: performance: linpack: x[n-1]-1 -1.30534172e-05
bright: h7: performance: linpack: smatgen 1 seconds 0.01083
bright: h7: performance: linpack: smatgen 2 seconds 0.01083
bright: h7: performance: linpack: Repetitions 82
bright: h7: performance: linpack: Leading dimension 201
bright: h7: performance: linpack: sdgefa sdgesl total Mflops
bright: h7: performance: linpack: 1 pass seconds 0.04000 0.01000 0.05000
bright: h7: performance: linpack: Repeat seconds 0.04954 0.00146 0.05100 13.46
bright: h7: performance: linpack: Repeat seconds 0.04966 0.00146 0.05112 13.43
bright: h7: performance: linpack: Repeat seconds 0.04966 0.00146 0.05112 13.43
bright: h7: performance: linpack: Repeat seconds 0.04966 0.00146 0.05112 13.43

bright: h7: performance: linpack: Repeat seconds 0.04966 0.00146 0.05112 13.43
bright: h7: performance: linpack: Average 13.44
bright: h7: performance: linpack: Leading dimension 200
bright: h7: performance: linpack: Repeat seconds 0.04917 0.00146 0.05064 13.56
bright: h7: performance: linpack: Repeat seconds 0.04929 0.00146 0.05076 13.53
bright: h7: performance: linpack: Repeat seconds 0.04917 0.00146 0.05064 13.56
bright: h7: performance: linpack: Repeat seconds 0.04929 0.00146 0.05076 13.53
bright: h7: performance: linpack: Repeat seconds 0.04917 0.00159 0.05076 13.53
bright: h7: performance: linpack: Average 13.54
bright: h7: performance: linpack: Finished test on Mon Dec 13 12:40:09 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h7
bright: performance: linpack: h8: Module started Tue Dec 14 12:40:07 EST 1999
bright: h8: performance: linpack: LINPACK BENCHMARK
bright: h8: performance: linpack: CPU
bright: h8: performance: linpack: Clock MHz
bright: h8: performance: linpack: Cache
bright: h8: performance: linpack: Rolling Unrolled
bright: h8: performance: linpack: Precision Single
bright: h8: performance: linpack: norm. resid 1.9
bright: h8: performance: linpack: resid 4.52336171e-05
bright: h8: performance: linpack: machep 1.19209290e-07
bright: h8: performance: linpack: x[0]-1 -1.31130219e-05
bright: h8: performance: linpack: x[n-1]-1 -1.30534172e-05
bright: h8: performance: linpack: smatgen 1 seconds 0.01087
bright: h8: performance: linpack: smatgen 2 seconds 0.01087
bright: h8: performance: linpack: Repetitions 81
bright: h8: performance: linpack: Leading dimension 201
bright: h8: performance: linpack: sdgefa sdgesl total Mflops
bright: h8: performance: linpack: 1 pass seconds 0.05000 0.00000 0.05000
bright: h8: performance: linpack: Repeat seconds 0.05036 0.00148 0.05184 13.25
bright: h8: performance: linpack: Repeat seconds 0.05024 0.00148 0.05172 13.28
bright: h8: performance: linpack: Repeat seconds 0.05036 0.00148 0.05184 13.25
bright: h8: performance: linpack: Repeat seconds 0.05024 0.00148 0.05172 13.28
bright: h8: performance: linpack: Repeat seconds 0.05036 0.00148 0.05184 13.25
bright: h8: performance: linpack: Average 13.26
bright: h8: performance: linpack: Leading dimension 200
bright: h8: performance: linpack: Repeat seconds 0.04950 0.00148 0.05098 13.47
bright: h8: performance: linpack: Repeat seconds 0.04962 0.00148 0.05110 13.44
bright: h8: performance: linpack: Repeat seconds 0.04950 0.00148 0.05098 13.47
bright: h8: performance: linpack: Repeat seconds 0.04962 0.00148 0.05110 13.44
bright: h8: performance: linpack: Repeat seconds 0.04950 0.00148 0.05098 13.47
bright: h8: performance: linpack: Average 13.46
bright: h8: performance: linpack: Finished test on Tue Dec 14 12:41:38 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h8
bright: performance: linpack: h9: Module started Tue Dec 14 01:15:37 EST 1999
bright: h9: performance: linpack: LINPACK BENCHMARK
bright: h9: performance: linpack: CPU
bright: h9: performance: linpack: Clock MHz
bright: h9: performance: linpack: Cache
bright: h9: performance: linpack: Rolling Unrolled
bright: h9: performance: linpack: Precision Single
bright: h9: performance: linpack: norm. resid 1.9
bright: h9: performance: linpack: resid 4.52336171e-05
bright: h9: performance: linpack: machep 1.19209290e-07
bright: h9: performance: linpack: x[0]-1 -1.31130219e-05
bright: h9: performance: linpack: x[n-1]-1 -1.30534172e-05

bright: h9: performance: linpack: smatgen 1 seconds 0.01089
bright: h9: performance: linpack: smatgen 2 seconds 0.01084
bright: h9: performance: linpack: Repetitions 80
bright: h9: performance: linpack: Leading dimension 201
bright: h9: performance: linpack: sdgefa sdgesl total Mflops
bright: h9: performance: linpack: 1 pass seconds 0.05000 0.01000 0.06000
bright: h9: performance: linpack: Repeat seconds 0.05098 0.00150 0.05248 13.08
bright: h9: performance: linpack: Repeat seconds 0.05098 0.00150 0.05248 13.08
bright: h9: performance: linpack: Repeat seconds 0.05098 0.00150 0.05248 13.08
bright: h9: performance: linpack: Repeat seconds 0.05098 0.00150 0.05248 13.08
bright: h9: performance: linpack: Repeat seconds 0.05098 0.00150 0.05248 13.08
bright: h9: performance: linpack: Average 13.08
bright: h9: performance: linpack: Leading dimension 200
bright: h9: performance: linpack: Repeat seconds 0.04953 0.00150 0.05103 13.46
bright: h9: performance: linpack: Repeat seconds 0.04941 0.00150 0.05091 13.49
bright: h9: performance: linpack: Repeat seconds 0.04941 0.00150 0.05091 13.49
bright: h9: performance: linpack: Repeat seconds 0.04953 0.00150 0.05103 13.46
bright: h9: performance: linpack: Repeat seconds 0.04941 0.00150 0.05091 13.49
bright: h9: performance: linpack: Average 13.48
bright: h9: performance: linpack: Finished test on Tue Dec 14 01:17:07 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h9
bright: performance: linpack: h10: Module started Mon Dec 13 12:25:54 EST 1999
bright: h10: performance: linpack: LINPACK BENCHMARK
bright: h10: performance: linpack: CPU
bright: h10: performance: linpack: Clock MHz
bright: h10: performance: linpack: Cache
bright: h10: performance: linpack: Rolling Unrolled
bright: h10: performance: linpack: Precision Single
bright: h10: performance: linpack: norm. resid 1.9
bright: h10: performance: linpack: resid 4.52336171e-05
bright: h10: performance: linpack: machep 1.19209290e-07
bright: h10: performance: linpack: x[0]-1 -1.31130219e-05
bright: h10: performance: linpack: x[n-1]-1 -1.30534172e-05
bright: h10: performance: linpack: smatgen 1 seconds 0.01084
bright: h10: performance: linpack: smatgen 2 seconds 0.01084
bright: h10: performance: linpack: Repetitions 81
bright: h10: performance: linpack: Leading dimension 201
bright: h10: performance: linpack: sdgefa sdgesl total Mflops
bright: h10: performance: linpack: 1 pass seconds 0.05000 0.01000 0.06000
bright: h10: performance: linpack: Repeat seconds 0.05051 0.00148 0.05200 13.21
bright: h10: performance: linpack: Repeat seconds 0.05051 0.00160 0.05212 13.17
bright: h10: performance: linpack: Repeat seconds 0.05039 0.00160 0.05200 13.21
bright: h10: performance: linpack: Repeat seconds 0.05051 0.00148 0.05200 13.21
bright: h10: performance: linpack: Repeat seconds 0.05051 0.00148 0.05200 13.21
bright: h10: performance: linpack: Average 13.20
bright: h10: performance: linpack: Leading dimension 200
bright: h10: performance: linpack: Repeat seconds 0.04940 0.00160 0.05101 13.46
bright: h10: performance: linpack: Repeat seconds 0.04953 0.00148 0.05101 13.46
bright: h10: performance: linpack: Repeat seconds 0.04953 0.00148 0.05101 13.46
bright: h10: performance: linpack: Repeat seconds 0.04953 0.00160 0.05113 13.43
bright: h10: performance: linpack: Repeat seconds 0.04940 0.00160 0.05101 13.46
bright: h10: performance: linpack: Average 13.46
bright: h10: performance: linpack: Finished test on Mon Dec 13 12:27:25 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h10
bright: performance: linpack: h11: Module started Mon Dec 13 13:13:56 EST 1999
bright: h11: performance: linpack: LINPACK BENCHMARK

```
bright: h11: performance: linpack: CPU
bright: h11: performance: linpack: Clock MHz
bright: h11: performance: linpack: Cache
bright: h11: performance: linpack: Rolling          Unrolled
bright: h11: performance: linpack: Precision        Single
bright: h11: performance: linpack: norm. resid            1.9
bright: h11: performance: linpack: resid            4.52336171e-05
bright: h11: performance: linpack: machep           1.19209290e-07
bright: h11: performance: linpack: x[0]-1           -1.31130219e-05
bright: h11: performance: linpack: x[n-1]-1         -1.30534172e-05
bright: h11: performance: linpack: smatgen 1 seconds         0.01083
bright: h11: performance: linpack: smatgen 2 seconds         0.01083
bright: h11: performance: linpack: Repetitions               83
bright: h11: performance: linpack: Leading dimension         201
bright: h11: performance: linpack:                  sdgefa     sdgesl     total     Mflops
bright: h11: performance: linpack: 1 pass seconds          0.05000    0.01000    0.06000
bright: h11: performance: linpack: Repeat seconds          0.04929    0.00145    0.05074     13.53
bright: h11: performance: linpack: Repeat seconds          0.04929    0.00157    0.05086     13.50
bright: h11: performance: linpack: Repeat seconds          0.04929    0.00145    0.05074     13.53
bright: h11: performance: linpack: Repeat seconds          0.04929    0.00145    0.05074     13.53
bright: h11: performance: linpack: Repeat seconds          0.04929    0.00145    0.05074     13.53
bright: h11: performance: linpack: Average                                       13.53
bright: h11: performance: linpack: Leading dimension         200
bright: h11: performance: linpack: Repeat seconds          0.04905    0.00145    0.05050     13.60
bright: h11: performance: linpack: Repeat seconds          0.04893    0.00145    0.05038     13.63
bright: h11: performance: linpack: Repeat seconds          0.04893    0.00145    0.05038     13.63
bright: h11: performance: linpack: Repeat seconds          0.04905    0.00145    0.05050     13.60
bright: h11: performance: linpack: Repeat seconds          0.04905    0.00145    0.05050     13.60
bright: h11: performance: linpack: Average                                       13.61
bright: h11: performance: linpack: Finished test on Mon Dec 13 13:15:27 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h11
bright: performance: linpack: h12: Module started Mon Dec 13 12:36:29 EST 1999
bright: h12: performance: linpack: LINPACK BENCHMARK
bright: h12: performance: linpack: CPU
bright: h12: performance: linpack: Clock MHz
bright: h12: performance: linpack: Cache
bright: h12: performance: linpack: Rolling          Unrolled
bright: h12: performance: linpack: Precision        Single
bright: h12: performance: linpack: norm. resid            1.9
bright: h12: performance: linpack: resid            4.52336171e-05
bright: h12: performance: linpack: machep           1.19209290e-07
bright: h12: performance: linpack: x[0]-1           -1.31130219e-05
bright: h12: performance: linpack: x[n-1]-1         -1.30534172e-05
bright: h12: performance: linpack: smatgen 1 seconds         0.01089
bright: h12: performance: linpack: smatgen 2 seconds         0.01083
bright: h12: performance: linpack: Repetitions               80
bright: h12: performance: linpack: Leading dimension         201
bright: h12: performance: linpack:                  sdgefa     sdgesl     total     Mflops
bright: h12: performance: linpack: 1 pass seconds          0.05000    0.00000    0.05000
bright: h12: performance: linpack: Repeat seconds          0.05111    0.00163    0.05273     13.02
bright: h12: performance: linpack: Repeat seconds          0.05111    0.00150    0.05261     13.05
bright: h12: performance: linpack: Repeat seconds          0.05111    0.00150    0.05261     13.05
bright: h12: performance: linpack: Repeat seconds          0.05111    0.00150    0.05261     13.05
bright: h12: performance: linpack: Repeat seconds          0.05123    0.00150    0.05273     13.02
bright: h12: performance: linpack: Average                                       13.04
bright: h12: performance: linpack: Leading dimension         200
```

bright: h12: performance: linpack: Repeat seconds 0.04892 0.00150 0.05042 13.62
bright: h12: performance: linpack: Repeat seconds 0.04905 0.00150 0.05055 13.58
bright: h12: performance: linpack: Repeat seconds 0.04905 0.00150 0.05055 13.58
bright: h12: performance: linpack: Repeat seconds 0.04892 0.00150 0.05042 13.62
bright: h12: performance: linpack: Repeat seconds 0.04905 0.00150 0.05055 13.58
bright: h12: performance: linpack: Average 13.60
bright: h12: performance: linpack: Finished test on Mon Dec 13 12:37:59 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h12
bright: performance: linpack: h13: Module started Mon Dec 13 13:47:10 EST 1999
bright: h13: performance: linpack: LINPACK BENCHMARK
bright: h13: performance: linpack: CPU
bright: h13: performance: linpack: Clock MHz
bright: h13: performance: linpack: Cache
bright: h13: performance: linpack: Rolling Unrolled
bright: h13: performance: linpack: Precision Single
bright: h13: performance: linpack: norm. resid 1.9
bright: h13: performance: linpack: resid 4.52336171e-05
bright: h13: performance: linpack: machep 1.19209290e-07
bright: h13: performance: linpack: x[0]-1 -1.31130219e-05
bright: h13: performance: linpack: x[n-1]-1 -1.30534172e-05
bright: h13: performance: linpack: smatgen 1 seconds 0.01087
bright: h13: performance: linpack: smatgen 2 seconds 0.01087
bright: h13: performance: linpack: Repetitions 82
bright: h13: performance: linpack: Leading dimension 201
bright: h13: performance: linpack: sdgefa sdgesl total Mflops
bright: h13: performance: linpack: 1 pass seconds 0.05000 0.00000 0.05000
bright: h13: performance: linpack: Repeat seconds 0.04998 0.00146 0.05144 13.35
bright: h13: performance: linpack: Repeat seconds 0.04998 0.00159 0.05156 13.32
bright: h13: performance: linpack: Repeat seconds 0.04998 0.00146 0.05144 13.35
bright: h13: performance: linpack: Repeat seconds 0.04998 0.00159 0.05156 13.32
bright: h13: performance: linpack: Repeat seconds 0.04998 0.00146 0.05144 13.35
bright: h13: performance: linpack: Average 13.34
bright: h13: performance: linpack: Leading dimension 200
bright: h13: performance: linpack: Repeat seconds 0.05022 0.00146 0.05169 13.29
bright: h13: performance: linpack: Repeat seconds 0.05034 0.00146 0.05181 13.25
bright: h13: performance: linpack: Repeat seconds 0.05022 0.00159 0.05181 13.25
bright: h13: performance: linpack: Repeat seconds 0.05022 0.00159 0.05181 13.25
bright: h13: performance: linpack: Repeat seconds 0.05022 0.00159 0.05181 13.25
bright: h13: performance: linpack: Average 13.26
bright: h13: performance: linpack: Finished test on Mon Dec 13 13:48:42 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h13
bright: performance: linpack: h14: Module started Mon Dec 13 13:37:08 EST 1999
bright: h14: performance: linpack: LINPACK BENCHMARK
bright: h14: performance: linpack: CPU
bright: h14: performance: linpack: Clock MHz
bright: h14: performance: linpack: Cache
bright: h14: performance: linpack: Rolling Unrolled
bright: h14: performance: linpack: Precision Single
bright: h14: performance: linpack: norm. resid 1.9
bright: h14: performance: linpack: resid 4.52336171e-05
bright: h14: performance: linpack: machep 1.19209290e-07
bright: h14: performance: linpack: x[0]-1 -1.31130219e-05
bright: h14: performance: linpack: x[n-1]-1 -1.30534172e-05
bright: h14: performance: linpack: smatgen 1 seconds 0.01081
bright: h14: performance: linpack: smatgen 2 seconds 0.01084
bright: h14: performance: linpack: Repetitions 83

bright: h14: performance: linpack: Leading dimension 201
bright: h14: performance: linpack: sdgefa sdgesl total Mflops
bright: h14: performance: linpack: 1 pass seconds 0.05000 0.01000 0.06000
bright: h14: performance: linpack: Repeat seconds 0.04943 0.00145 0.05087 13.50
bright: h14: performance: linpack: Repeat seconds 0.04931 0.00145 0.05075 13.53
bright: h14: performance: linpack: Repeat seconds 0.04931 0.00157 0.05087 13.50
bright: h14: performance: linpack: Repeat seconds 0.04931 0.00145 0.05075 13.53
bright: h14: performance: linpack: Repeat seconds 0.04931 0.00145 0.05075 13.53
bright: h14: performance: linpack: Average 13.52
bright: h14: performance: linpack: Leading dimension 200
bright: h14: performance: linpack: Repeat seconds 0.04952 0.00157 0.05108 13.44
bright: h14: performance: linpack: Repeat seconds 0.04952 0.00145 0.05096 13.47
bright: h14: performance: linpack: Repeat seconds 0.04952 0.00145 0.05096 13.47
bright: h14: performance: linpack: Repeat seconds 0.04952 0.00157 0.05108 13.44
bright: h14: performance: linpack: Repeat seconds 0.04952 0.00145 0.05096 13.47
bright: h14: performance: linpack: Average 13.46
bright: h14: performance: linpack: Finished test on Mon Dec 13 13:38:40 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h14
bright: performance: linpack: h15: Module started Mon Dec 13 12:43:00 EST 1999
bright: h15: performance: linpack: LINPACK BENCHMARK
bright: h15: performance: linpack: CPU
bright: h15: performance: linpack: Clock MHz
bright: h15: performance: linpack: Cache
bright: h15: performance: linpack: Rolling Unrolled
bright: h15: performance: linpack: Precision Single
bright: h15: performance: linpack: norm. resid 1.9
bright: h15: performance: linpack: resid 4.52336171e-05
bright: h15: performance: linpack: machep 1.19209290e-07
bright: h15: performance: linpack: x[0]-1 -1.31130219e-05
bright: h15: performance: linpack: x[n-1]-1 -1.30534172e-05
bright: h15: performance: linpack: smatgen 1 seconds 0.01081
bright: h15: performance: linpack: smatgen 2 seconds 0.01081
bright: h15: performance: linpack: Repetitions 83
bright: h15: performance: linpack: Leading dimension 201
bright: h15: performance: linpack: sdgefa sdgesl total Mflops
bright: h15: performance: linpack: 1 pass seconds 0.05000 0.00000 0.05000
bright: h15: performance: linpack: Repeat seconds 0.04931 0.00157 0.05087 13.50
bright: h15: performance: linpack: Repeat seconds 0.04931 0.00145 0.05075 13.53
bright: h15: performance: linpack: Repeat seconds 0.04931 0.00157 0.05087 13.50
bright: h15: performance: linpack: Repeat seconds 0.04931 0.00145 0.05075 13.53
bright: h15: performance: linpack: Repeat seconds 0.04931 0.00157 0.05087 13.50
bright: h15: performance: linpack: Average 13.51
bright: h15: performance: linpack: Leading dimension 200
bright: h15: performance: linpack: Repeat seconds 0.04871 0.00145 0.05015 13.69
bright: h15: performance: linpack: Repeat seconds 0.04871 0.00157 0.05027 13.66
bright: h15: performance: linpack: Repeat seconds 0.04871 0.00145 0.05015 13.69
bright: h15: performance: linpack: Repeat seconds 0.04871 0.00145 0.05015 13.69
bright: h15: performance: linpack: Repeat seconds 0.04871 0.00157 0.05027 13.66
bright: h15: performance: linpack: Average 13.68
bright: h15: performance: linpack: Finished test on Mon Dec 13 12:44:31 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h15
bright: performance: linpack: h16: Module started Mon Dec 13 13:05:47 EST 1999
bright: h16: performance: linpack: LINPACK BENCHMARK
bright: h16: performance: linpack: CPU
bright: h16: performance: linpack: Clock MHz
bright: h16: performance: linpack: Cache

bright: h16: performance: linpack: Rolling Unrolled
bright: h16: performance: linpack: Precision Single
bright: h16: performance: linpack: norm. resid 1.9
bright: h16: performance: linpack: resid 4.52336171e-05
bright: h16: performance: linpack: machep 1.19209290e-07
bright: h16: performance: linpack: x[0]-1 -1.31130219e-05
bright: h16: performance: linpack: x[n-1]-1 -1.30534172e-05
bright: h16: performance: linpack: smatgen 1 seconds 0.01083
bright: h16: performance: linpack: smatgen 2 seconds 0.01081
bright: h16: performance: linpack: Repetitions 82
bright: h16: performance: linpack: Leading dimension 201
bright: h16: performance: linpack: sdgefa sdgesl total Mflops
bright: h16: performance: linpack: 1 pass seconds 0.05000 0.00000 0.05000
bright: h16: performance: linpack: Repeat seconds 0.04990 0.00159 0.05149 13.34
bright: h16: performance: linpack: Repeat seconds 0.04990 0.00146 0.05137 13.37
bright: h16: performance: linpack: Repeat seconds 0.04990 0.00146 0.05137 13.37
bright: h16: performance: linpack: Repeat seconds 0.05003 0.00146 0.05149 13.34
bright: h16: performance: linpack: Repeat seconds 0.04990 0.00146 0.05137 13.37
bright: h16: performance: linpack: Average 13.36
bright: h16: performance: linpack: Leading dimension 200
bright: h16: performance: linpack: Repeat seconds 0.04907 0.00146 0.05053 13.59
bright: h16: performance: linpack: Repeat seconds 0.04907 0.00146 0.05053 13.59
bright: h16: performance: linpack: Repeat seconds 0.04894 0.00146 0.05041 13.62
bright: h16: performance: linpack: Repeat seconds 0.04907 0.00146 0.05053 13.59
bright: h16: performance: linpack: Repeat seconds 0.04907 0.00146 0.05053 13.59
bright: h16: performance: linpack: Average 13.60
bright: h16: performance: linpack: Finished test on Mon Dec 13 13:07:18 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h16
bright: performance: linpack: h2: Module started Mon Dec 13 17:51:10 EST 1999
bright: h2: performance: linpack: LINPACK BENCHMARK
bright: h2: performance: linpack: CPU
bright: h2: performance: linpack: Clock MHz
bright: h2: performance: linpack: Cache
bright: h2: performance: linpack: Rolling Unrolled
bright: h2: performance: linpack: Precision Single
bright: h2: performance: linpack: norm. resid 1.9
bright: h2: performance: linpack: resid 4.52336171e-05
bright: h2: performance: linpack: machep 1.19209290e-07
bright: h2: performance: linpack: x[0]-1 -1.31130219e-05
bright: h2: performance: linpack: x[n-1]-1 -1.30534172e-05
bright: h2: performance: linpack: smatgen 1 seconds 0.01086
bright: h2: performance: linpack: smatgen 2 seconds 0.01084
bright: h2: performance: linpack: Repetitions 81
bright: h2: performance: linpack: Leading dimension 201
bright: h2: performance: linpack: sdgefa sdgesl total Mflops
bright: h2: performance: linpack: 1 pass seconds 0.05000 0.00000 0.05000
bright: h2: performance: linpack: Repeat seconds 0.05050 0.00148 0.05198 13.21
bright: h2: performance: linpack: Repeat seconds 0.05050 0.00160 0.05210 13.18
bright: h2: performance: linpack: Repeat seconds 0.05038 0.00160 0.05198 13.21
bright: h2: performance: linpack: Repeat seconds 0.05050 0.00148 0.05198 13.21
bright: h2: performance: linpack: Repeat seconds 0.05050 0.00160 0.05210 13.18
bright: h2: performance: linpack: Average 13.20
bright: h2: performance: linpack: Leading dimension 200
bright: h2: performance: linpack: Repeat seconds 0.04891 0.00148 0.05039 13.63
bright: h2: performance: linpack: Repeat seconds 0.04891 0.00148 0.05039 13.63
bright: h2: performance: linpack: Repeat seconds 0.04891 0.00148 0.05039 13.63

```
bright: h2: performance: linpack: Repeat seconds          0.04891   0.00148   0.05039    13.63
bright: h2: performance: linpack: Repeat seconds          0.04891   0.00148   0.05039    13.63
bright: h2: performance: linpack: Average                                     13.63
bright: h2: performance: linpack: Finished test on Mon Dec 13 17:54:14 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h2
bright: performance: linpack: h3: Module started Mon Dec 13 12:35:51 EST 1999
bright: h3: performance: linpack: LINPACK BENCHMARK
bright: h3: performance: linpack: CPU
bright: h3: performance: linpack: Clock MHz
bright: h3: performance: linpack: Cache
bright: h3: performance: linpack: Rolling         Unrolled
bright: h3: performance: linpack: Precision       Single
bright: h3: performance: linpack: norm. resid             1.9
bright: h3: performance: linpack: resid           4.52336171e-05
bright: h3: performance: linpack: machep          1.19209290e-07
bright: h3: performance: linpack: x[0]-1          -1.31130219e-05
bright: h3: performance: linpack: x[n-1]-1        -1.30534172e-05
bright: h3: performance: linpack: smatgen 1 seconds       0.01086
bright: h3: performance: linpack: smatgen 2 seconds       0.01084
bright: h3: performance: linpack: Repetitions             82
bright: h3: performance: linpack: Leading dimension       201
bright: h3: performance: linpack:                 sdgefa    sdgesl    total    Mflops
bright: h3: performance: linpack: 1 pass seconds          0.05000   0.00000   0.05000
bright: h3: performance: linpack: Repeat seconds          0.04975   0.00159   0.05134    13.38
bright: h3: performance: linpack: Repeat seconds          0.04975   0.00146   0.05121    13.41
bright: h3: performance: linpack: Repeat seconds          0.04987   0.00146   0.05134    13.38
bright: h3: performance: linpack: Repeat seconds          0.04987   0.00146   0.05134    13.38
bright: h3: performance: linpack: Repeat seconds          0.04987   0.00146   0.05134    13.38
bright: h3: performance: linpack: Average                                     13.38
bright: h3: performance: linpack: Leading dimension       200
bright: h3: performance: linpack: Repeat seconds          0.04940   0.00146   0.05086    13.50
bright: h3: performance: linpack: Repeat seconds          0.04940   0.00159   0.05099    13.47
bright: h3: performance: linpack: Repeat seconds          0.04940   0.00146   0.05086    13.50
bright: h3: performance: linpack: Repeat seconds          0.04940   0.00146   0.05086    13.50
bright: h3: performance: linpack: Repeat seconds          0.04940   0.00146   0.05086    13.50
bright: h3: performance: linpack: Average                                     13.49
bright: h3: performance: linpack: Finished test on Mon Dec 13 12:37:22 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h3
bright: performance: linpack: h4: Module started Mon Dec 13 12:21:38 EST 1999
bright: h4: performance: linpack: LINPACK BENCHMARK
bright: h4: performance: linpack: CPU
bright: h4: performance: linpack: Clock MHz
bright: h4: performance: linpack: Cache
bright: h4: performance: linpack: Rolling         Unrolled
bright: h4: performance: linpack: Precision       Single
bright: h4: performance: linpack: norm. resid             1.9
bright: h4: performance: linpack: resid           4.52336171e-05
bright: h4: performance: linpack: machep          1.19209290e-07
bright: h4: performance: linpack: x[0]-1          -1.31130219e-05
bright: h4: performance: linpack: x[n-1]-1        -1.30534172e-05
bright: h4: performance: linpack: smatgen 1 seconds       0.01081
bright: h4: performance: linpack: smatgen 2 seconds       0.01081
bright: h4: performance: linpack: Repetitions             83
bright: h4: performance: linpack: Leading dimension       201
bright: h4: performance: linpack:                 sdgefa    sdgesl    total    Mflops
bright: h4: performance: linpack: 1 pass seconds          0.05000   0.01000   0.06000
```

bright: h4: performance: linpack: Repeat seconds 0.04931 0.00145 0.05075 13.53
bright: h4: performance: linpack: Repeat seconds 0.04943 0.00145 0.05087 13.50
bright: h4: performance: linpack: Repeat seconds 0.04931 0.00145 0.05075 13.53
bright: h4: performance: linpack: Repeat seconds 0.04931 0.00157 0.05087 13.50
bright: h4: performance: linpack: Repeat seconds 0.04931 0.00145 0.05075 13.53
bright: h4: performance: linpack: Average 13.52
bright: h4: performance: linpack: Leading dimension 200
bright: h4: performance: linpack: Repeat seconds 0.04943 0.00145 0.05087 13.50
bright: h4: performance: linpack: Repeat seconds 0.04943 0.00145 0.05087 13.50
bright: h4: performance: linpack: Repeat seconds 0.04931 0.00157 0.05087 13.50
bright: h4: performance: linpack: Repeat seconds 0.04931 0.00145 0.05075 13.53
bright: h4: performance: linpack: Repeat seconds 0.04943 0.00145 0.05087 13.50
bright: h4: performance: linpack: Average 13.50
bright: h4: performance: linpack: Finished test on Mon Dec 13 12:23:09 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h4
bright: performance: linpack: h5: Module started Mon Dec 13 19:24:38 EST 1999
bright: h5: performance: linpack: LINPACK BENCHMARK
bright: h5: performance: linpack: CPU
bright: h5: performance: linpack: Clock MHz
bright: h5: performance: linpack: Cache
bright: h5: performance: linpack: Rolling Unrolled
bright: h5: performance: linpack: Precision Single
bright: h5: performance: linpack: norm. resid 1.9
bright: h5: performance: linpack: resid 4.52336171e-05
bright: h5: performance: linpack: machep 1.19209290e-07
bright: h5: performance: linpack: x[0]-1 -1.31130219e-05
bright: h5: performance: linpack: x[n-1]-1 -1.30534172e-05
bright: h5: performance: linpack: smatgen 1 seconds 0.01087
bright: h5: performance: linpack: smatgen 2 seconds 0.01086
bright: h5: performance: linpack: Repetitions 81
bright: h5: performance: linpack: Leading dimension 201
bright: h5: performance: linpack: sdgefa sdgesl total Mflops
bright: h5: performance: linpack: 1 pass seconds 0.05000 0.00000 0.05000
bright: h5: performance: linpack: Repeat seconds 0.05085 0.00160 0.05246 13.09
bright: h5: performance: linpack: Repeat seconds 0.05073 0.00160 0.05233 13.12
bright: h5: performance: linpack: Repeat seconds 0.05085 0.00148 0.05233 13.12
bright: h5: performance: linpack: Repeat seconds 0.05085 0.00148 0.05233 13.12
bright: h5: performance: linpack: Repeat seconds 0.05085 0.00148 0.05233 13.12
bright: h5: performance: linpack: Average 13.11
bright: h5: performance: linpack: Leading dimension 200
bright: h5: performance: linpack: Repeat seconds 0.05013 0.00148 0.05161 13.30
bright: h5: performance: linpack: Repeat seconds 0.05013 0.00148 0.05161 13.30
bright: h5: performance: linpack: Repeat seconds 0.05013 0.00148 0.05161 13.30
bright: h5: performance: linpack: Repeat seconds 0.05013 0.00148 0.05161 13.30
bright: h5: performance: linpack: Repeat seconds 0.05013 0.00148 0.05161 13.30
bright: h5: performance: linpack: Average 13.30
bright: h5: performance: linpack: Finished test on Mon Dec 13 19:26:10 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h5
bright: performance: linpack: h6: Module started Mon Dec 13 12:41:43 EST 1999
bright: h6: performance: linpack: LINPACK BENCHMARK
bright: h6: performance: linpack: CPU
bright: h6: performance: linpack: Clock MHz
bright: h6: performance: linpack: Cache
bright: h6: performance: linpack: Rolling Unrolled
bright: h6: performance: linpack: Precision Single
bright: h6: performance: linpack: norm. resid 1.9

```
bright: h6: performance: linpack: resid             4.52336171e-05
bright: h6: performance: linpack: machep            1.19209290e-07
bright: h6: performance: linpack: x[0]-1           -1.31130219e-05
bright: h6: performance: linpack: x[n-1]-1         -1.30534172e-05
bright: h6: performance: linpack: smatgen 1 seconds          0.01085
bright: h6: performance: linpack: smatgen 2 seconds          0.01086
bright: h6: performance: linpack: Repetitions                82
bright: h6: performance: linpack: Leading dimension          201
bright: h6: performance: linpack:                  sdgefa    sdgesl    total    Mflops
bright: h6: performance: linpack: 1 pass seconds          0.05000   0.00000   0.05000
bright: h6: performance: linpack: Repeat seconds          0.04988   0.00146   0.05135    13.37
bright: h6: performance: linpack: Repeat seconds          0.04988   0.00146   0.05135    13.37
bright: h6: performance: linpack: Repeat seconds          0.04988   0.00146   0.05135    13.37
bright: h6: performance: linpack: Repeat seconds          0.04988   0.00146   0.05135    13.37
bright: h6: performance: linpack: Repeat seconds          0.04988   0.00146   0.05135    13.37
bright: h6: performance: linpack: Average                                      13.37
bright: h6: performance: linpack: Leading dimension          200
bright: h6: performance: linpack: Repeat seconds          0.04987   0.00146   0.05133    13.38
bright: h6: performance: linpack: Repeat seconds          0.04999   0.00146   0.05145    13.35
bright: h6: performance: linpack: Repeat seconds          0.04987   0.00159   0.05145    13.35
bright: h6: performance: linpack: Repeat seconds          0.04987   0.00146   0.05133    13.38
bright: h6: performance: linpack: Repeat seconds          0.04987   0.00159   0.05145    13.35
bright: h6: performance: linpack: Average                                      13.36
bright: h6: performance: linpack: Finished test on Mon Dec 13 12:43:19 EST 1999
bright: Deleteing temporary file /home/dmattoon/.bright/bright-linpackpoerf.h6
bright: performance: netpipe: Executed netpipe.
bright: Executing performance-nested user-defined tests (if any)
bright: All tests completed Mon Dec 13 20:02:09 1999
```

REFERENCES

1. Barnes, Donnie. *RPM Howto (RPM at idle)*. http://www.rpm.org/support/RPM-HOWTO.html

2. Becker, Donald J., John Salmon, Daniel F. Savarese, Thomas L. Sterling. *How to Build A Beowulf: A Guide to the Implementation and Application of PC Clusters*. The MIT Press. Cambridge, Massachusetts. 1999.

3. Begeulin, Adam, Jack Dongarra, Al Geist, Wiecheng Jiang, Robert Manchek, Vaidy Sunderam. *PVM: Parallel Virtual Machine - A Users Guide and Tutorial for Networked Parallel Computing*. The MIT Press. Cambridge, Massachusetts. 1994.

4. Gropp, William, Ewing Lusk, Anthony Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. The MIT Press. Cambridge, Massachusetts. 1994.
5. Helmer, Guy. *NETPIPE: A Network Protocol Independent Performance Evaluator*. http://www.scl.ameslab.gov/netpipe. January 14, 1998.

6. Merkey, Phillip. *Introduction to the CESDIS Beowulf Project*. http://beowulf.gsfc.nasa.gov/intro.html

7. Muuss, Mike. *The Story of the PING program*. http://ftp.arl.mil/~mike/ping.html.

8. Sill, Dave. *Benchmarks FAQ.* http://hpwww.epfl.ch/bench/bench.FAQ.html.

9. Packet Storm Security. *Index of UNIX Utilities.* http://packetstorm.security.com/UNIX/utilities.